2007

IVEE: Interesting video event extraction

Jeremy Paskali

IVEE: Interesting Video Event Extraction

Jeremy C. Paskali

Department of Computer Science
Rochester Institute of Technology

January 2007

A thesis, submitted to the faculty of the Golisano College of Computing and Information Sciences in partial fulfillment of the requirement for the degree of Master of Science in Computer Science.

Chair: Prof. Roger Gaborski Date

Reader: Prof. Carl Reynolds Date

Observer: Prof. Zack Butler Date

Abstract

The Video Exploitation and Novelty Understanding in Streams (VENUS) system is a complete software solution for video surveillance, which consists of motion detection, motion tracking, novel event detection, missed expected event detection, object recognition, scene description, and database mining. This research focuses on the assignment of degrees of interest for events in motion and at rest for the VENUS system. The Interesting Video Event Extraction (IVEE) system is the second module in the processing pipeline of the VENUS system. The novel features of the IVEE system include the ability to assign a degree of interest to an event, to develop a representation of the weekly activities from the input video stream, and to detect when an expected event did not occur. The IVEE system maintains independence through self-learning and without the aid of human intervention to understand the difference between normal and abnormal behaviors.

Acknowledgments

My first acknowledgments go to my parents, Paula Mae Cook and Anthony Paskali, and to my grandparents, Pauline Cook and Everett Cook for supporting me in life, in school, and being there for me at all times. I also give acknowledgments to my graduate committee, Professor Roger Gaborski as the chair, Professor Carl Reynolds as the reader, and Professor Zack Butler as the observer for providing the education and insight I needed to complete my Masters degree at RIT. More acknowledgments go to the great students and professors in the computer vision lab and around the computer science department, including Professor Hans-Peter Bischof, Professor Joe Geigel, T.J. Borrelli, Karthik Chandrasekaran, John Lareau, Dave Wollenhaupt, and Michael Mertsock for all their support and intellectual conversations.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

The goal of this research is to classify the degree of interest of events with respect to previous events in a video stream. The degree of interest in an event is associated with the similarity of the event in relation to previously witnessed events. The higher the similarity between a new event and witnessed events, the lower the interest in the new event. People may find similarities amongst events via comparisons of the colors, shapes, textures, sounds, tastes, smells, velocities, sizes, locations, and much more. As people become insensitive to similar input stimuli, they are able to focus on novel stimuli. By focusing on the novel stimuli, people can be aware of dangerous situations and changes in the surrounding environment.

The problem contains two sources of input data. The first source is the input video stream, which consists of a constant width and height, a defined number of frames, a constant frames per second, and red, green, and blue pixel intensity data planes. The second source of input is the motion coordinates data, which is created in Matlab and stored in a standard Matlab matrix file.

1.2 VENUS

VENUS is an acronym for Video Exploitation and Novelty Understanding in Streams. The first generation VENUS system was designed, implemented, and reported by Gaborski *et al.* [10]. Unfortunately, the first generation

VENUS system was lost and never rebuilt. Instead of rebuilding the first generation VENUS system, a second generation VENUS system will be built as a replacement and upgrade.

The first generation VENUS system concentrated on using saliency maps to focus on regions of interest in the input video streams. David Burlone enhanced the detection of saliency in video streams through his research and implementation [6]. Although David Burlone's work may become a part of the second generation VENUS system, the current research on VENUS is headed away from saliency and toward the relation of the interest amongst events.

The term, interesting, describes an event in the input video stream and is a combination of both novelty and missed expected event detection. Novelty event detection is the search for objects in motion or at rest that have been forgotten or have never been witnessed before. People often forget specific events and therefore will consider that event novel if witnessed again. However, novelty event detection does not include the surprise due to a missed occurrence of an expected periodic event. This is where missed expected event detection describes the rest of the term, interesting, which is the search for the absence of objects in motion or at rest that are expected to occur at specific times.

The second generation VENUS system is being built with consideration to code modularity. By maintaining loosely coupled modules in the VENUS system, a researcher can easily remove an outdated module and replace it with an enhanced module. For instance, a researcher designs and implements a more efficient algorithm for the motion and tracking module, then that researcher can replace the module without worrying about any consequences to other modules. In fact, the only consequence out of this process would be faster running times and more accurate results. This upgrade process for the second generation VENUS system will aid in the system's lifetime as researchers tackle various sub-problems within VENUS.

1.3 IVEE

IVEE is an acronym, which stands for Interesting Video Event Extraction. The IVEE system is a module that assigns the degree of interest for events in the input video stream for the second generation VENUS system. The IVEE system obtains input from the motion and tracking module, which delivers

the original input video stream and the motion coordinates data structure. The output of the IVEE system includes the interesting event coordinates data structure, the interesting event output video stream, the memory of interesting events, and the missed expected events list.

The quality of the IVEE system's output depends heavily upon the motion coordinates data structure provided by the motion and tracking module. Clearly, if the motion and tracking module declares many false positive or false negative events in motion and if the tracking is not performed properly, then the IVEE system will be directly relying on this inaccurate data. This leads to the simple strategy that if any module is ever upgraded in the VENUS system, then that module's output should be at least as accurate and precise as the previous module's output.

The IVEE system contains motion and still interesting subsystems. The motion interesting subsystem assigns interest values to events in motion, whereas the still interesting subsystem assigns interest values to still events. To exemplify this, a car driving on a road is an event in motion, whereas a car that has just parked is a still event. The two subsystems will run separately, which will allow for the analysis of still events separately from motion events and for the replacement of either of the subsystems without affecting the other.

1.4 Theory Overview

The overall design strategy of this research is to build membership groups based on low-level event features extracted from the input video stream. Each witnessed event is compared to local groups, which represent the system's memory of specific types of events. The system assigns an interest value based upon the similarity of the event to the existing groups. If the event is similar enough, which is defined by a confidence percentage, then the group's means and variances for each feature are updated based upon the event features. Furthermore, the group membership level is increased, which is a representation of the system's familiarity with these specific types of events. If the event is not similar enough to any groups, then a new group is created to represent the witnessed event. However, the system assigned event interest level is still based upon the closest membership group match.

Every group contains a mean and a variance for each extractable low-level feature of events from the input video stream. The feature list includes

red pixel intensity, green pixel intensity, blue pixel intensity, pixel height, pixel width, horizontal pixel velocity, vertical pixel velocity, and the hours from the beginning of Sunday when the event occurred during the week. Furthermore, groups consist of a membership level, the last date when the group was updated by a witnessed event, the last date when the group was updated by a witnessed event or partially forgotten by the IVEE system, and the last interest value assigned to a witnessed event by this group.

Groups are associated with a specific eight by eight pixel region, or patch, of the input video stream. This division of regions allows for spatial separation of events across the input video stream. By spatially separating groups across discrete regions, common events in specific regions will be considered uncommon, thus interesting, outside those regions. As an example, if students walk along a sidewalk all day and the IVEE system learns this as normal, the system would still find students walking on the grass as interesting because it occurs outside the sidewalk regions. The eight by eight pixel regions along the sidewalk contain high group membership levels for students walking, whereas the regions of the grass contain no group information or low group membership levels for students walking. Similar to how a normal event is only normal in relation to the surrounding environment, like an airplane flying in the sky versus driving on a highway, the IVEE system maintains this concept through the distinct regions of groups.

Each group contains one membership level, which is a representation of the IVEE system's familiarity with a specific type of event. The higher the membership level, the more familiar the IVEE system is with the group. Similarly, the higher the membership level, the IVEE system will find any similar events less interesting. In order for a group to exist, the group membership level must be greater than zero, but the IVEE system places no limit on the ceiling for the membership level. No upper limit on the membership level allows for the representation of the IVEE system to be extremely familiar with events that occur on a weekly basis. Furthermore, if IVEE is extremely familiar with a particular event, then it will take many weeks to completely forget that event due to the large group membership level.

For each group in the IVEE system's memory, a set of low-level input video stream features describes the events experienced and integrated into the group. Each group feature contains one mean and one variance, which are initially set upon creation of the group and are updated upon discovering a similar witnessed event. As Figure 1.1 shows, each feature is plotted against the group membership level to obtain a feature membership triangle. The

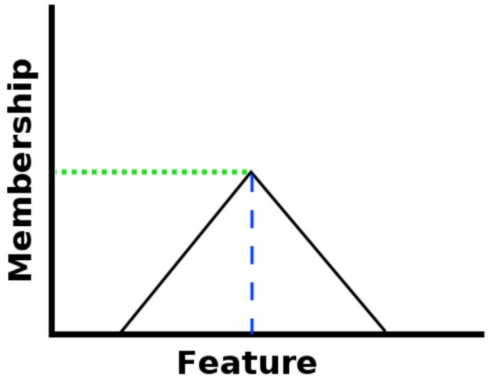

Figure 1.1: The peak of the feature membership triangle is determined by the feature mean and the group membership level. The triangle width is twice the feature variance.

height of the triangle is the group membership level and the width is twice the feature variance. In order for the IVEE system to assign interest values to witnessed events, each feature of the event falls somewhere on the edge of the feature membership triangle, which may fall below zero. The closer each event feature is to the group feature mean, the more similar the witnessed event is to the group. The user defines a percentage confidence value, which is a threshold for determining if the witnessed event belongs to the group.

1.5 Implementation Overview

The Motion/Still Interest module is the main entry point for the IVEE system. This module initializes the major data structures and runs the main loop for extracting the frames of the input video stream. The most important data structure, which will be referred to as the IVEE system's memory, holds the group information across all eight by eight pixel regions. The IVEE system's memory allows for the relational assignment of interest values to witnessed events based on experienced events. Each witnessed event is only compared to groups of the same region of the IVEE system's memory. Although an event will often be large enough to encompass more than one region, the calculated pixel center of the event will determine which region the event will be compared against groups. The IVEE system assigns

an interest value to the witnessed event based upon the comparisons of the regional groups.

The Motion/Still Interest module maps the assigned event interest value to a color. The IVEE system displays a rectangle of the mapped color about the event on the output video stream. The assigned event interest value is inversely proportional to the value of the hue plane in the Hue-Saturation-Value (HSV) color space. The value and saturation planes are held constant at one. Therefore, an interest value of one would map to a hue value of zero, which would display as a red rectangle about the event on the output video stream. As Figure 5.3 shows, the hue value increases, or moves around the circle counter-clockwise, as the interest value decreases. However, assigned interest values above the user-defined threshold will have rectangles drawn about them of the proper color and assigned interest values below this threshold will not have rectangles drawn about them. Events with rectangles not drawn about them are either normal or expected events.

The Motion/Still Interest module calls upon the Missed Groups module, which determines when an expected event did not occur. An expected event arises when the same event occurs week after week and the IVEE system eventually expects that this event should occur with an expected set of features. If the event does not occur within the expected time, the Missed Groups module writes the group feature means and variances to a missed expected events text file and copies the group's location and size to a data structure. This data structure is used to draw where the missed expected event should have taken place in the output video stream.

The Motion/Still Interest module also calls upon the Forget Group module, which determines if a group should be partially or fully forgotten. If a group should be forgotten, the Forget Group module subtracts the proper amount of membership from the group. If the membership reaches or falls below zero, the group is removed from the IVEE system's memory. The user defines the frequency of the forget period, which is compared against the last time a group was updated and the current date of the input video stream. For every forget period that fits between the group update date and the current date, the Forget Group module subtracts from the group membership.

The Motion/Still Interest module also calls upon the Merge Groups module, which determines if two groups should be merged into a single group. The IVEE system uses one of either two criteria to determine if the groups should be merged. The first criterion is that the first group's feature means have a higher membership in the second group than in itself, or vice versa.

The second criterion is that the first group's feature means pass a user-defined similarity confidence value to the second group and vice-versa. If either one of these two criteria are satisfied, then the groups will be merged.

1.6 Thesis Outline

The next chapter of this thesis report contains definitions to establish a baseline of understanding before the reader proceeds. Chapter 3 contains three sections of background and related work material. Section 3.1 discusses the biological factors involved with novelty, surprise, and forming memories. Section 3.2 surveys the current state of the art in the surveillance technology domain in academia. Section 3.3 reviews the original VENUS work and follows with a modular description of the current VENUS work. Chapter 4 surveys the possible mechanisms to solving the problem of relating interest amongst events. Chapter 5 details the implementation of the IVEE system and lists the requirements. Chapter 6 enumerates and discusses the experiments performed on the IVEE system. Finally, possible future work and closing remarks for the IVEE system are discussed.

Chapter 2

Definitions

2.1 Event Types

2.1.1 Normal Events

For people, normal events involve input stimuli that are not expected, but also not noticeably different from the usual input stimuli. For instance, a person driving a vehicle along a highway would not take special notice of vehicles on the other side of the barrier driving in the opposite direction. That person does not have an expectation of the size, make, model, and color of those vehicles, the person just understands that vehicles travel in that given direction and contain features within certain variances.

For events in motion to be considered normal, the features of that event, including color, size, velocity, time, and location, must be similar enough to previously experienced events. Furthermore, in order for events to be considered normal, yet not expected, there should be no pattern developed or developing in the time domain. As an example, students walking across a college campus are a normal event because the features are similar amongst the students. However, if a student happens to dress a specific way for each day of the week, week after week, and commits to walking a specific path to class for each day of the week, one would come to expect this behavior throughout the semester. This last example is clearly not a normal event anymore, but an expected event.

Stationary events can also be considered normal. A normal stationary event occurs when the background of a scene is altered by a motion event coming to rest in that scene. As an example, if a camera is placed to look

at open lockers in an elementary school, then the lockers will be filled with bags, lunches, and jackets in the morning and then emptied in the afternoon. People would both not expect specific features of the objects going into the open lockers and not be interested in the fact that the lockers are being filled during the day.

2.1.2 Interesting Events

Interesting events consist of features extremely different from previously witnessed events. For people, the features of the event are different enough to cause them to give attention to that particular event. For instance, a person driving a vehicle on a highway notices vehicles in the rear view mirror, but does not pay special attention to them because it is a normal occurrence. However, when a police officer turns on the flashing lights of the police vehicle, the person will give the event special attention because it is interesting. The interesting part is the intensity, color, and strobing effect features of the police vehicle's lights.

Interesting events also include missed expected events. If a person expects a safety officer to drive by the dormitory every hour on the hour and this event does not occur within a given expected time frame, then the person is interested in the fact that the usually recurring event did not occur.

The same concepts for interesting events in motion apply to stationary events. An example of an interesting stationary event would be a person placing a backpack near a trash can in a mall and walking away. Another example would be a vehicle parked in a restricted parking area. The IVEE system will consider anything that becomes stationary and usually is not stationary at that location interesting.

2.1.3 Expected Events

Eventually, a person will consider an event that was interesting to now be normal because the event has been continuing to take place. Furthermore, that person may start to notice a pattern in the occurrence of the normal event, which would lead the person to expect the event at specific times. Once the person expects the event to take place at a specific time and the event does not occur, the person will be alarmed at the absence of the event.

2.1.4 Forgetting

If an expected event continues to not occur in the expected time frames, a person would stop expecting the event to occur and think of the event as normal again, but not associated with any schedule. If the event continues to not occur at all, the person would be interested if the event did occur. Eventually, the person would consider the event novel because the person has not witnessed the event in such a long time.

2.2 Schedules

The majority of working people follow a typical weekly schedule. To clarify, the schedule is typical for the individual and not for everybody. Person A may work Monday to Friday from 9:00 AM to 5:00 PM, while person B may work on Monday, Tuesday, Thursday, Friday, and Saturday from 8:00 AM to 4:00 PM. Person A expects to arrive at work just before 9:00 AM just as much as Person B expects to arrive at work before 8:00 AM. Furthermore, the management at the workplaces expects the people to arrive on time for work. Expectations develop quickly from people following daily and weekly routines.

Weekly schedules are critical to the life of a business. Employees are expected to arrive and to leave at given time intervals to be able to interact amongst each other. Business deliveries and shipments usually are processed at specific times during the day. In fact, most businesses expect no activity in and around the business facilities at given times, such as weekends, late night weekdays, and early morning weekdays. Many expectations are built up from simple work schedules.

Chapter 3

Background/Related Work

3.1 Biological Considerations

3.1.1 Semantics

James and Gauthier designed and ran a study to determine the regions of the brain that handle semantics of everyday objects on an involuntary basis. They conducted a small study where two groups of six participants were trained to correctly recognize at least twenty-two of twenty-four objects, of which some are shown in Figure 3.1. The first group of participants were trained with two sets of twelve objects, where each set was of the color blue. The participants had to match objects to names or adjectives within three seconds for all twenty-four objects. The second group of participants were presented with two sets of twelve objects as well, where both sets had a unique color. The second group of participants were allowed four seconds to correctly match objects to names or adjectives. The first group had four letter long male nicknames to learn, whereas the second group had a first, middle, and last name to learn. The first set of objects were associated with adjectives along with the names and referred to as the SEM set. The second set of objects were associated with a name only and referred to as the NAM set. The final set of objects, with which the participants were not trained, had no textual associations with them and were referred to as the NON set [14]. This would allow the study of the brain's reaction to semantic information attached to the object.

The study produced some interesting findings with regards to brain stimulation. The left inferior frontal and parietal cortex were activated greater

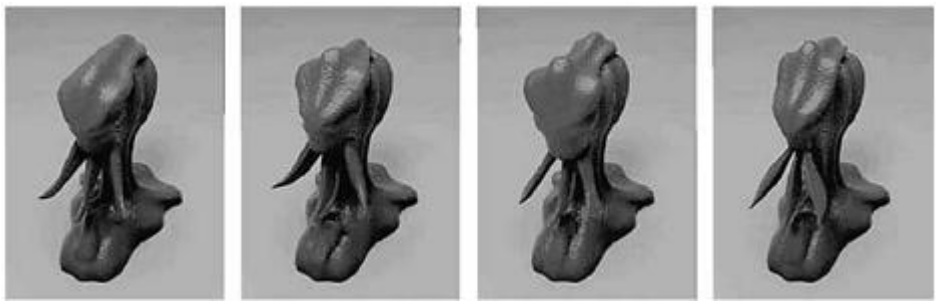

Figure 3.1: The objects that participants matched to adjectives or names varied slightly in appearance [14].

with the SEM set than with the NON set. Furthermore, only participants from the second group caused an extremely high amount of activation in the post-central gyrus. The NON objects elicited a higher activation of the occipital brain region than the SEM objects most likely because participants had to focus more on the physical traits of the object and not rely on the learned textual associations. The brain activity difference between the NON objects and the NAM objects were minimal. In fact, the only difference was in the medial temporal lobe, which was more active in the NON objects. The authors stated that this was most likely because the NON objects were more novel and that names alone truly do not provide the same information as adjectives [14]. For example, a person can associate the adjective cold with snow, ice, and winter. However, a person usually does not associate Mark or William Phillip Smith with other ideas.

James and Gauthier made conclusions based on the study findings. First, the left inferior frontal cortex definitely processes semantic information of objects. Furthermore, this region of the brain is known to remember arbitrary associations learned a long time ago and recently. Objects associated with semantical information appeared to hinder the participants' abilities to correctly associate the object with the set. The left inferior frontal cortex appears to process semantic information throughout its entirety. Furthermore, it appears to store learned associations in a modality-specific way. It was also noted that associations learned verbally are stored differently than when learned via visual sensors [14].

3.1.2 Novelty Seeking

In a long-term study, Laucht *et al.* performed an experiment on infants, and continued intermittently up to fifteen years of age, which would help link genes to an individual's desire for seeking novelty. This trait tends to make people distracted, perhaps impulsive, and even disorderly in their lives. Previous studies have had inconsistent outcomes when determining if this novelty seeking gene varies with the dopamine D4 receptor (DRD4) in the brain. It has also been known that females do not show as much novelty seeking as males [17].

The study contains 232 infants born between 1986 and 1988. They are then assigned to a group depending on their degree of perinatal and psychosocial risk. The children had to be the firstborn, where they were the only child at that birth, the parents had to speak German, and the children had to have no major handicaps, genetic defects, or diseases. The humans were studied at three months, two, four and a half, five, eight, eleven, and fifteen years of age [17].

Infants were placed in front of a monitor, which displayed a doll's face for the first six trials as the non-novel stimulus, whereas the last two trials were a picture of a checkerboard and a teddy bear as the novel stimuli. If a child looked at the stimulus for longer than one second, that child was considered to be fixated on the stimulus. Similarly, if the child looked away from the stimulus for longer than one second, then the child was considered to break that fixation. While no relationships could be determined with the girls, the boys that looked away from the repeated non-novel stimulus quicker and were fixated on the novel stimuli longer tended to show novelty seeking behaviors more as teenagers. Laucht *et al.* conclude that the boys with this behavior more often have the DRD4 7r allele, whereas boys without this allele do not seek novelty as much [17].

3.1.3 Entropy and Surprise

Strange *et al.* created a study to measure the hippocampal response to entropy within a sequence of events. The general thought is that if a brain region is more active when something seems incorrect, out of place, or surprising in an event stream, then this area of the brain must be responsible for error prediction. Entropy is defined as the average expectation of events in a stream. On the other hand, surprise is the strangeness of a single event

in a stream [29].

It appears the left anterior hippocampus is activated more when the event streams have high entropy. The more the entropy, the more the activation. However, the anterior hippocampus does not alter in activation from any surprising events. Surprising events lead to activation in the bilateral network. Thus, it is noted that entropy and surprise occur in distinct regions in the brain [29].

Strange *et al.* conclude that the hippocampal contains knowledge of the probability of when surprising events occur, but not the actual experiences. This is seen when repeated surprising events change the activation of the anterior hippocampus. The basic idea here is that the hippocampus is a person's representation of what he expects to occur next with events [29].

3.1.4 Extraneous Activity

Kiehl *et al.* discovered activation within regions of the brain through the projection of images to participants. Participants would be shown 244 stimuli throughout the test. Three categories of stimuli exist: target, novel, and non-target. The target stimulus was a small white square, the novel stimuli were shapes of various colors, and the non-target stimulus was a large white square. Each stimulus had a probability of appearing next in the sequence of shapes. The participants' goal is to respond as quickly as possible when the small white box was shown. The main outcome from this experiment is the fact that the brain often creates activity in brain regions outside of those regions that should be the only regions to handle the signal. By providing extra activity to extraneous brain regions, the stimuli might be better processed beyond just what the current generation of people can perform. In this way, it seems as though it is an evolutionary strategy for the brain to activate more regions of the brain then necessary to complete a task [15].

3.1.5 Temporal Memories

Previous studies show that the hippocampus allows for the creation of memories and allows for the recall of memories within a context of time in a person's life. Furthermore, this brain region allows for three dimensional mapping of the environment. The experiment performed by Huerta *et al.* measures a rat's ability to remember a Pavlovian association between a conditioned stimulus, white noise, and an unconditioned stimulus, electric shock.

Three sets of rats exist in this experiment: control, mutant, and pseudo. The control rats were normal, healthy rats. The mutants were missing a specific receptor in the NMDA receptors, which is located in the hippocampus. The pseudo rats were normal rats, but during the Pavlovian training they did not receive the unconditioned stimulus [12].

The experiment conditioned the control and mutant rats that when white noise is heard in the cage, that a shock will follow a few seconds later. This would cause the rats to stand still after they heard the white noise in fear of the coming electric shock. It took the mutant rats longer to learn the association. Furthermore, a test was conducted twenty-four hours after the Pavlovian association, which would test whether the rats remembered the association. The control rats froze significantly more than the mutants after hearing the white noise. The pseudo group of rats froze significantly less than the control rats, indicating that they have not associated the white noise with the electric shock. This shows that the control group truly did learn the association. This experiment showed that the NMDA receptors in the hippocampus are required to form temporal memories [12].

3.2 State of the Art

3.2.1 On-line Novelty Detection

Novelty detection is often times implemented for mobile robots, which use cameras to determine the status of the environment, the location of the robot, and if there is a change in the environment. Neto and Nehmzow compared Grow-When-Required (GWR) artificial neural networks versus Principal Component Analysis (PCA). Both methodologies must be able to run in an on-line environment. Furthermore, the images were split into twenty-four by twenty-four pixel patches, which were used with saliency maps in order to reduce the total amount of image data to be processed. The top nine most salient patches were selected to discover novel objects [22].

The GWR networks initialize with two unconnected neurons. The neuron weights are set to the first input data. Afterward, the input data is compared against the neuron weights and if the difference exceeds a threshold then the input data is considered novel. Furthermore, the network will grow to represent the novel data so that if it does see this input data again it will not be considered novel. The GWR network builds up a topological representation

of the input space [22].

The PCA algorithm would not normally be able to run in an on-line environment. However, Neto and Nehmzow implemented an incremental PCA algorithm, developed by Artac *et al.*, which is able to run in an on-line environment [1]. With this algorithm, the amount of error between the data and the reconstruction of that data onto the eigenspace determines if the input is novel [22].

The mobile robot uses a basic algorithm to avoid obstacles and traverse the environment while observing the environment with the camera. The environment is then altered and the robot is set loose to see if it can discover the difference. The robot uses the saliency map on the fly to determine the top nine most salient regions and then compares those regions using both the algorithms to see if it is novel. Neto and Nehmzow conclude that the PCA has an advantage over GWR because it has slightly better performance and a better ability to reconstruct the original data after compression, but it does require more memory than GWR [22].

3.2.2 Artificial Eye Scanning

Ban and Lee designed and implemented a biologically inspired system for detecting novelty in a dynamic environment. The system can handle slight affine transformations with the camera and still recognize when a novel event occurs. The system is built from a five-layer hierarchical fuzzy Adaptive Resonance Theory (ART) neural network and a bottom-up saliency map. The system requires a human supervisor to show the system which areas are important and which areas are not important in an environment. This information is stored within the layers of the ART network. The top network layer stores the most abstract information, whereas the bottom layer stores detailed information. The system uses the most salient areas along with the ART network to create a scan path, which is similar to how one's eye might move through any particular environment. The system calculates energy signatures across the regions of the scan path and then calculates the Euclidean distance difference across consecutive video frames. Anything that passes a given threshold is considered novel [3].

3.2.3 Spatiotemporal Blocks

Pokrajac *et al.* created a solution for bypassing noisy video streams for surveillance systems. They break the video streams into spatiotemporal blocks of size eight by eight by three. These blocks represent both the textures and the motion in the video. The algorithm has two major phases. The first phase is to reduce the dimensions in the spatiotemporal blocks through the use of PCA. The second phase is to detect movement in the blocks through a modified version of incremental Expectation-Maximization (EM) algorithm, which allows for the estimation of Gaussian mixtures. The components contain a mean, diagonal covariance matrix, and a distributional prior. Using the squared Mahalanobis distance, a block is only considered to be in motion if the minimum distance is above a set threshold [25].

This algorithm was tested against additive Gaussian noise, which had a zero mean with variances ranging from 0.01 to 0.5, salt and pepper noise, which had a density ranging from 0.05 to 0.2, multiplicative or speckle noise, which ranged from 0.01 to 0.5, and finally Poisson noise. The analysis showed that the algorithm correctly identified moving objects throughout the video stream in all cases. The authors conclude that their algorithm is better suited for noise than algorithms based on pixel movement detection [25].

3.2.4 Abnormal Behavior Detection

Dahmane and Meurnier created a real-time video surveillance system that uses self-organizing maps (SOMs) to detect abnormal behavior in a scene. The major components of their system include the detection of moving objects, tracking those objects, and the classification of behavior as normal or abnormal. The authors used a background subtraction method, which utilizes a nonparametric statistical technique to determine the moving objects in the scene. The most interesting part of the background model is removing shadow detection. A shadow greatly drops the intensity value of a pixel, but the Red-Green-Blue (RGB) components of the pixel remain in the same ratio. Basically, Dahmane and Meurnier developed a subsystem that looks to see if a pixel has become darker, but the chromatic distortion has been minimal. If this is true, then it is most likely that pixel is a shadow. However, for each scene, the threshold for determining if a shadow exists must be hand tuned by a human [9].

The system maintains a background model by first updating a parallel

background model. The background model, which is used to subtract from the current scene, is updated periodically from the parallel background model only where no motion has occurred. Furthermore, the authors decided to use a three by three median filter across the image to reduce any camera fluctuations and thus reduce noises, especially about edges in the scene [9].

Dahmane and Meurnier implemented a procedure that matches the tracking of direct and inverse matrices. To simplify tracking, they utilize a bounding box for each blob in the scene. Blobs are considered to be merging when the centroid of one enters the bounding box of another. After merging, blobs often split apart again. In order for the system to identify entities during the split correctly, the system maintains color information for each individual blob. The system monitors each moving object over its lifetime in the video scene. The tracking algorithm understands objects merging, splitting, entering, leaving, and blob correspondence [9].

In order to detect abnormal behavior, the system examines two types of behavior. The first is the local behavior, which is simply an object's position and instantaneous velocity. Second, the global behavior is basically the flow of the object in the scene, which is described by elliptic Fourier descriptors. In order to analyze these two types of behavior models, two SOMs are implemented, one for local and one for global behaviors. The SOMs must first be trained to learn the normal behavior in a particular video scene. Secondly, the SOMs enter the operational phase where current behavior vectors are compared against the winning neurons in the SOM. If a threshold is surpassed, then the behavior is considered to be abnormal. Dahmane and Meurnier achieved a 94.6% correct classification rate. Furthermore, their system runs with a minimum of twelve hertz on a 2.66 gigahertz processor [9].

3.2.5 Audio-based Surveillance

Clavel *et al.* designed an audio-based surveillance system, which could be incorporated into a video surveillance system, thus providing another dimension of data to aid the system. The system built by Clavel *et al.* only handles gun shot detection, but it could be extended. The two major focuses of the system are the ability to handle non-optimal conditions and reduce all false positives and negatives as much as possible [8].

The shot detection system is tested with both a two class system and a four class system. The two class system contains a shot and normal class,

of which both are modeled with Gaussian Mixture Models (GMMs). Sound samples are taken in twenty millisecond clips, with neighbor overlap of fifty percent. Each vector has features including volume, the first eight Mel-Frequency Cepstral Coefficients, first two spectral statistical moments (mean of power spectrum and spectral spread), and the first two derivatives of the mentioned features. The dimensions are reduced by using PCA [8].

The system was trained with 134 shots of various weapons and public places recordings. The training depends heavily on the sound to noise ratio (SNR): the greater the energy of the shot with respect to the environment energy, the better the quality of finding gun shots during testing. However, perfectly clean shot databases often lead to more false rejections for noisy test sequences. On the other hand, noisy shot training databases caused a higher level of false acceptance. The optimal SNR was twenty decibels for training, which caused a false rejection rate of less than eleven percent and a false acceptance rate of less than fifteen percent. The use of hierarchical classification with placing similar sounding gun shots in their own classes helped the system. Using three gun classes with one normal class caused a false rejection rate of ten percent and a false detection rate of less than five percent [8].

3.3 VENUS

3.3.1 First Generation System

System Overview

Gaborski *et al.* designed, implemented, and tested the first generation Video Exploitation and Novelty Understanding in Streams (VENUS) System, which included both motion and still novelty detection. The core areas of VENUS comprised of saliency, focus of attention, and data mining. Saliency and focus of attention work together to reduce the amount of processing. Motion and still saliency maps provide the areas within the video for the system to focus. Thus, the system can focus on the objects with the most contrast, in terms of low-level features, instead of the entire scene [10].

For each eight by eight pixel region, the low-level features of salient objects are compared against an existing pool of clusters. A Gaussian mixture model enables the identification of novel clusters for each region. The system

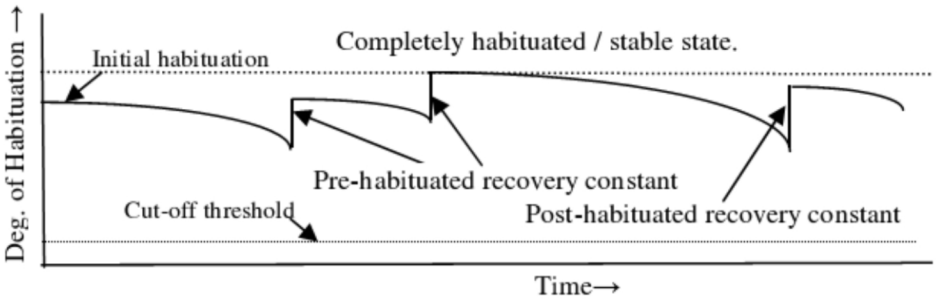

Figure 3.2: The degree of habituation decreases at a quicker rate as the current time progresses further away from the time the cluster was updated [10].

associates a habituation value for each cluster. The habituation function for a cluster determines the habituation value for a given time step:

$$H\left(t\right) = 1 - \left[1/\left(1 + e^{-at}\right)\right] \tag{3.1}$$

Here, $H(t)$ represents the habituation value after t frames have passed since the creation of the cluster. The decay rate of the cluster is represented by a. The decay rate is dynamic and follows the following formula:

$$a_t = 1 - [e/t] \tag{3.2}$$

The decay rate after t frames of the creation of the cluster is represented by a_t. e represents this cluster's merge count with similar clusters. Figure 3.2 shows an example of habituation over time [10].

Saliency

A recent extension of the VENUS system was the addition of saliency maps to detect salient objects in video streams. Burlone used intensity, orientation, yellow-blue, and red-green filters to determine saliency in each of these areas. These individual saliency maps are summed together to achieve the final saliency map for the image in the video. Burlone's results show that attention focuses on objects that appear to be most disconnected with their surrounding environments [6].

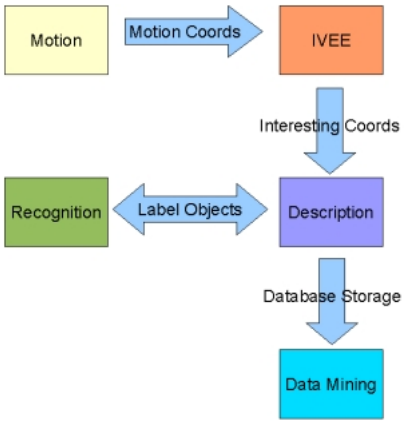

Figure 3.3: The VENUS modules interact together by saving and loading data structures to the disk. This creates a loosely coupled system that enables each module to be replaced without adverse effects.

System Shortcomings

Although the VENUS system is powerful for short term video sequences, it lacks knowledge about time and the absence of habituated events. The first generation VENUS system does not incorporate any time domain knowledge. If the decay rate is set fast enough, an event that occurs every weekday at 5:00 PM would never be habituated because it occurs too infrequently. However, a person would eventually habituate this weekly event, while not expecting it to occur on the weekends. Furthermore, that same person would become suspicious if that event did not occur on a weekday at 5:00 PM. Just as novel detection is important, so is the detection of the absence of recurring events. The system should also be able to handle events that occur on some sort of schedule. By the nature of human beings and our surrounding environment, people live on some sort of schedule, which causes a certain amount of expectations and regularity in people's lives.

3.3.2 Second Generation System

System Overview

The goal of the second generation VENUS system is to detect and track moving events, calculate the degree of interest in each event, label the interesting objects, describe the interesting scenes, and allow all the interesting video

sequences to be data mined. Currently, the Motion Detection and Interesting Event modules are built. Thus, the current VENUS system pipeline begins at the Motion Detection module and ends at the Interesting Event module. The completion date of the entire second generation VENUS system is unknown at this time.

Figure 3.3 depicts the module interaction in the VENUS system. This research only includes the Interesting Event module, which is also referred to as the IVEE system, and none of the other modules. This chapter provides the reader with the overall goals, strategies, and inter-communications amongst the VENUS system modules. The implementation and theoretical details of the IVEE system are discussed in Chapter 5.

Motion Detection Module

The Motion Detection module is the first step in the VENUS system pipeline. The module takes the raw video data as input and produces the motion maps and tracking data as output. The height and width of the motion maps are the same as the input video stream and a motion map must exist for each frame. At each location, the motion map may contain a zero or any positive integer. A motion map location with a zero represents no motion for the associated pixel coordinate of the specific frame in the input video stream. This leaves positive integers to represent pixel coordinates in motion. However, connected regions of movement, which would be associated with an event in motion, must be of the same positive integer value. Furthermore, all regions of movement in any frame must have unique positive integer values. Events in motion are tracked by maintaining the same unique positive integer values across multiple frames. As a final restriction, the positive integer assigned to an event in motion may be used again, but only after one buffer frame, where the positive integer does not appear. It is recommended that new integers should be used instead of repeating previously used integers.

An example will help clarify these rules for the motion maps and tracking data. A bus is in motion from frames seven to eighty in an input video stream. The bus is assigned the value of eighteen in frame seven. Thus, the bus must also be assigned the value of eighteen in frames eight through eighty. Furthermore, no other moving events in frames six through eighty-one can be assigned the value of eighteen. Here, frame six and frame eighty-one are the buffer frames needed before reusing the value eighteen. If a motorcycle moves from frame eighty-two to the end of the input video stream, it can

be assigned the value of eighteen. However, it is recommended that each event is assigned an unused value when possible. This can be done by simply creating a counter that increments with each new event in motion to keep track of the values assigned previously.

Interesting Event Module

The current Interesting Event module, which this research focuses on, is called the Interesting Video Event Extraction (IVEE) system. This system requires the motion map and tracking data from the Motion Detection module and the video stream as input. As an optional input, the IVEE system is able to accept its own memory data, which is also an output of the system. This additional input allows the IVEE system to maintain the knowledge of what the system has already learned from previous iterations. If this optional input is not present, the IVEE system starts with new memory, which contains no history of witnessed events.

The Interesting Event module outputs several important sets of data, which will be used by subsequent modules in the VENUS system pipeline. The first data set is a series of images, which represents the interesting events that occurred in the input video stream. The images of the interesting event are buffered by at least a user-defined number of images that do not contain interesting events, which is depicted in Figure 3.4. The second data set is the memory of interesting events for each eight by eight pixel region. The third data set is the interest data structure, which contains a logical motion map of the event location, the time of the event, and the interest level. Each data structure may contain zero or more instances of interesting events and there is one data structure for each frame of the input video stream. The final data set is a text file, which contains the features of all missed expected events that have occurred in the input video stream.

Scene Description Module

The Scene Description module will place metadata labels on the events, which will allow people to understand what occurred during the event time frame and will also allow the event to become indexed and searchable within a database. In order to accurately describe the interesting events, the Scene Description module will pass the interesting event motion maps to the Object Recognition module, which will identify the objects in the interesting

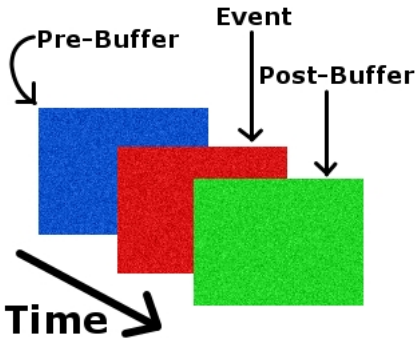

Figure 3.4: The frames in the video that contain the interesting event (in red) are buffered by a user-defined number of seconds of video both before (in blue) and after (in green) the event.

events and save the labels of objects to disk. Thus, the Scene Description module will gather the individual object labels along with metadata from the scene analysis algorithm to completely describe the scene with the interesting events. Input factors for the scene analysis include object labels, time of the day, day of the week, event locations, number of events, event interest levels, duration of events, and velocity of events. The metadata labels and the Interesting Event module output video will be sent to the Database module.

Object Recognition Module

The Object Recognition module will interact directly with the Scene Description module. The Object Recognition module will identify objects within events of interest. The label placed on each event will be sent back to the Description Module. Labels may include such objects as a person, car, airplane, deer, cow, etc. This module will most likely be modified and enhanced to discover objects within the environments of the specific input video streams. For instance, if a group of input video streams are of a parking lot, then the Object Recognition module will be tuned to identify people, cars, trucks, vans, police cars, fire trucks, and similar objects. This module could also be refined to identify subclasses of objects. For example, the Object Recognition module might be adjusted to identify whether the person is an employee or a visitor of an organization.

Database Module

The Database module will receive the labels from the Scene Description module. The Database module will correctly index the interest output video based upon the labels. Users will interact with the Database module to look up interesting events. For instance, a user will be able to search for events in specific locations of the video and with a given range of interest levels. A user will also be able to search based on keywords and time of day; such as 'person' between 2:00 AM and 4:00 AM. The Database module will return with the appropriate videos, if any exist. The Database module will also provide organizations with the ability to ignore video of normal activity, while only retaining video of unusual, suspicious, or abnormal activity.

Chapter 4

Relating Interest Amongst Events

4.1 Introduction

A major logistical problem with completing the task of assigning an interest value for each event is being able to correctly relate the interest to previously witnessed events. First, possible methodologies must be researched and reviewed for potential in solving this problem. Second, many possible methodologies must be narrowed down to a few based on performance, accuracy, precision, and the researchers ability to understand and implement the methodology. Finally, those few methodologies should be implemented, tested, and reviewed, which will allow the researcher to pick one to implement as the final solution.

The following sections, with exception to Section 4.4, are reprinted from the technical report, "Relating Interest Amongst Observed Events," with the permission from the Chair, Professor Roger Gaborski, and explore the various methodologies that have the potential to relate interest amongst observed events [23]. Section 4.4 details the methodology implemented in the IVEE system as the final solution.

4.2 Clustering

The first attempt at solving the problem of calculating an interest value involved clustering. Various clustering methods were analyzed, such as k-

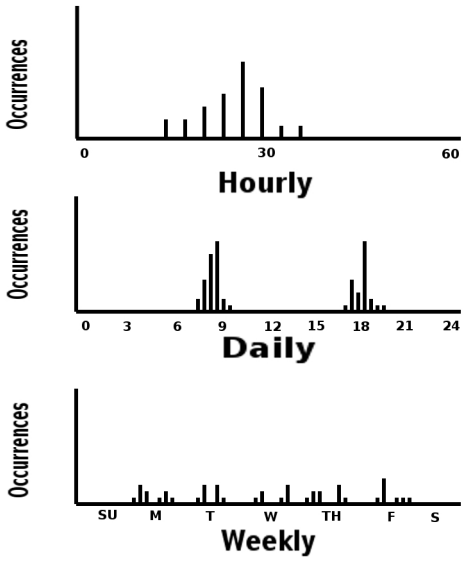

Figure 4.1: Observations are recorded with respect to many time scales. The challenge is to discover event patterns over one or more of these many time scales.

means, Bisecting k-means [7], Global k-means [18], Genetic k-means [16], X-means [24], G-means [11], and hierarchical clustering techniques.

Clustering techniques allow for the grouping and classification of data in high dimensional spaces. Selecting the best, or even an appropriate, clustering technique can be daunting and challenging when considering the vast number of existing algorithms. Not only do many mainstream clustering techniques exist, but there are also many variations.

Clustering techniques allow the IVEE system to solve the problem of relating interest amongst observed events by measuring differences amongst data points. Each defined cluster contains a center (centroid) and a variance, which enables the IVEE system to calculate the difference between the latest data point and the existing clusters. There are various methods to define the difference amongst data points, such as squared Euclidean distance, Manhattan distance, cosine, and correlation [20].

As depicted in Figure 4.1, every time a specific event occurs, like a red fire truck driving left to right across the video display, the time is recorded. Depending on the time, a normally uninteresting event could become an

interesting one. As an example, employee vehicles are expected in the parking lot typically from 6:00 AM to 8:00 PM. It might be slightly interesting if a person parks a vehicle at 9:00 PM. On the other hand, it should be extremely interesting if a person parks a vehicle at 1:00 AM due to the timing of the event with respect to a daily schedule. Furthermore, if employees typically park Monday through Friday only, any parking on Saturday and Sunday should be seen as interesting with respect to a weekly schedule. The answer to discovering if a normally uninteresting event is interesting due to the timing of the event may exist within clustering techniques. If the data clusters according to the timing of the event, as well as other measurables, then it may be possible to calculate the interest value based on all the information of the event, which includes the timing.

4.2.1 K-means

K-means clustering is an efficient algorithm with most data sets because it runs linearly, $O(n)$, on average [28] [13]. However, Vassilvitskii and Arthur have proven that the algorithm takes superpolynomial time, $2^{\Omega(\sqrt{n})}$, as a worst-case [2]. Furthermore, Pelleg and Moore claim k-means performs proportionately worse as the data set increases [24].

Sometimes the number of clusters is known a priori, which means k-means should be a great choice for clustering. Due to the nature of the data, the IVEE system does not know the exact number of clusters when calculating interest values for observed events. However, it is possible to approximate the range of values, but this method may also be inaccurate, which will lead to poorly chosen interest values. For instance, the daily time scale from Figure 4.1 may contain one to twenty-four clusters. However, there is the possibility of more than twenty-four clusters. Furthermore, it is difficult to understand when it is appropriate to increment to the next k value instead of stopping with the current k cluster setup.

K-means has a difficult time achieving the global minimum and often settles at a local minimum instead due to the initial positions of the centroids [13]. K-means is completely determinable given the initial centroid positions [24]. Placing the centroids in specific initial places for clustering may work for some events, but most likely not all. If the k-means algorithm finishes with a local minimum, then the IVEE system will assign events incorrect interest values. Bradley and Fayyad propose a method to enhance the initial positions of the centroids, but only offer a "better" local minimum and not

the global minimum [5].

4.2.2 Bisecting K-means

Bisecting k-means adds a layer around the standard k-means algorithm. It first calls k-means with k being equal to one. Then it selects either the largest cluster or the cluster with the most variance for splitting. Bisecting k-means calls k-means again with k being equal to two with the data set being equal to the chosen cluster to be split. This process repeats $k - 1$ times and the algorithm generates k clusters for the total data set, which do not overlap. Unlike the standard k-means algorithm, bisecting k-means is randomized, which means different clusters may result over multiple runs [7].

Cimiano, Hotho and Staab report better overall performance of bisecting k-means than standard k-means and agglomerative clustering techniques. In fact, bisecting k-means runs $O(nk)$ on average, where k is the number of desired clusters. At first glance, this may seem worse than the standard k-means algorithm, but k-mean's performance drops relatively quicker than the increase in the data set size [7]. Instead, bisecting k-means uses the standard k-means algorithm on smaller groups of data, which helps balance k-means inability to scale well with the size of the data set.

Bisecting k-means performs faster and with equal or better results than both standard k-means and Unweighted Pair Group Method with Arithmetic mean (UPGMA), which is an agglomerative hierarchical clustering technique [28]. It seems surprising that by simply wrapping k-means with another layer creates equivalent or better results than k-means and UPGMA. The major difficulty with bisecting k-means is determining an appropriate k value beforehand. Due to the nature of the data collection, the IVEE system will not know the appropriate number of clusters a priori.

4.2.3 Global K-means

Lika, Vlassis and Verbeek proposed the global k-means algorithm to provide a non-parametric and global minimum solution. Like the standard k-means algorithm, global k-means is deterministic. In fact, global k-means uses k-means with incremental k values, which allows global k-means to add one cluster at a time to achieve the global minimum solution [18].

The global k-means algorithm starts by calling the standard k-means algorithm with a k value of one. Each call of k-means is able to see the

entire data set, which is not true with the bisecting k-means algorithm. The k value is incremented and the optimal centroid position(s) are saved from the previous iteration. The global k-means algorithm calls the standard k-means algorithm for each point in the data set. The previous optimum $k-1$ centroids remain the same as the new k^{th} centroid starts at each point in the data set. The best solution is saved as the optimum solution to be used in the next iteration of the value of k. The entire process is repeated until the best value of k is discovered by using the process proposed by Milligan and Cooper [21].

The global k-means algorithm has the advantages of being non-parametric and guaranteed to find the optimum solution. Since the IVEE system does not know the number of clusters a priori, the non-parametric feature is a must. However, the run time of global k-means, even when adapted to run faster, runs ten to twenty times slower than k-means for a specific number of clusters [18]. It may be worth while to implement an algorithm that runs faster, is non-parametric, but might find a local minimum solution. This really depends on if the runtime of the IVEE system becomes an issue.

4.2.4 X-means

Pelleg and Moore proposed the X-means algorithm to combat the poor scalability of the standard k-means algorithm. Furthermore, the X-means algorithm requires only a range of possible number of clusters instead of an exact cluster number with k-means. Although the X-means algorithm does not guarantee the global minimum solution, it has a better chance than the standard k-means algorithm. On the other hand, X-means provides a much more scalable solution than k-means as the data sets increase in size by using caching techniques [24].

The X-means algorithm begins by running the standard k-means algorithm with the lower k value of the range provided as input. Each parent centroid is split into two child centroids. Again, k-means is called with a k value of two for each pair of children. The k-means works with the local cluster data set only, which helps overcome the scalability issue with k-means. A test is run to determine whether the children or the parent centroid better fit the data. The winner of the test remains while the loser is eliminated. The X-means algorithm continues to the next value of k until the upper value of the input range is exceeded [24].

The X-means algorithm may provide the middle ground between the stan-

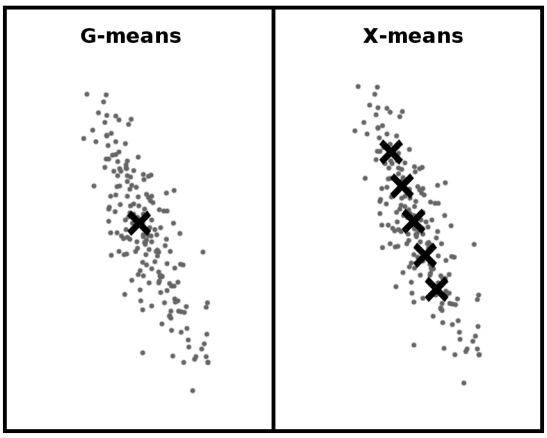

Figure 4.2: G-means accurately discovers cluster centroids with Gaussian distributed data sets, whereas X-means produces several false positives.

dard k-means and the global k-means algorithms. The X-means algorithm is more scalable than the k-means algorithm, yet should run faster than the global k-means algorithm. The global k-means algorithm searches through a more extensive range of k values to obtain a guaranteed global minimum, whereas the X-means algorithm only searches through the input range of k values to obtain a solution, which has better odds of being the global minimum when compared against k-means. The IVEE system may be able to provide an estimate range of k values as input to the X-means algorithm. The scalability of the algorithm is important because the data measured from observed events will quickly become large due to the many interesting events that could take place.

4.2.5 G-means

If the data observed from the events are believed to be Gaussian, a solution created by Hamerly and Elkan may prove best. Hamerly and Elkan call their algorithm G-means, which assumes the data points follow a Gaussian distribution. The key to this algorithm is maintaining cluster centers where the surrounding data points fit a Gaussian distribution and splitting the centers that do not. Hamerly and Elkan show that their algorithm finds clusters more precisely and with fewer errors than X-means [11].

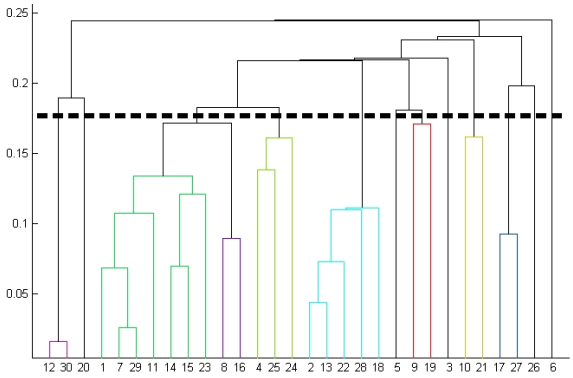

Figure 4.3: With agglomerative clustering, the user must decide upon a threshold, which directly impacts the number of clusters. In this example, the horizontal dashed line represents the threshold. If this line is raised, smaller clusters merge to form larger clusters, which reduces the total number of clusters across the data set.

G-means does not require user input of how many clusters are in the data set. Instead, it takes the standard statistical significance level, which defines a confidence measure of how much the data points follow a Gaussian distribution. If the statistical test shows that the local data points around a centroid follow a Gaussian distribution with the given input confidence level, then that centroid is kept, otherwise it is split into two and the algorithm runs the standard k-means algorithm on the local data points with a k value of two [11].

At this point, it is unknown if the data measured from observed events in the IVEE system follows a Gaussian model. However, the event data should be tested and if it is believed to follow a Gaussian model, then the G-means algorithm will be an excellent choice. At least with the example data sets shown by Hamerly and Elkan, the G-means algorithm outperforms X-means by finding a better centroid match to the data [11].

4.2.6 Agglomerative

The agglomerative algorithm is a type of hierarchical clustering technique. The goal of the algorithm is to build clusters from the bottom up, starting by assuming each data point is a cluster. From here, clusters are joined together based on a criterion, which is usually single-linkage, minimum-

variance or complete-linkage. The most popular methods are single-linkage and complete-linkage. Single-linkage looks for the minimum distances between points in clusters, whereas complete-linkage looks for the maximum distances between all pairs of points in each cluster. The clusters continue to be joined together until a threshold is reached for the chosen criterion, at which point the algorithm stops with the given cluster setup. The resulting clusters will most likely end up looking more like a string with single-linkage or a dense cloud with complete-linkage [13].

The agglomerative algorithm runs $O(n^3)$ naively, $O(n^2 \log n)$ optimized, and $O(n^2)$ with single-linkage as the criterion [7]. This average run time is much worse than the standard k-means algorithm and other variations. Often times, the efficiency of agglomerative clustering relies on the properties of the data set. For instance, agglomerative clustering does not work well with grouping documents due to the appearance of common words found in all documents. Because of this property of many languages, agglomerative clustering often ends up placing unlike documents in the same cluster [28].

Considering agglomerative clustering's weakness for similarities in data sets, the IVEE system will most likely not be implementing this technique. The data measured from observed events will contain many similarities. For each event, there are only three planes of color data, a medium range of velocities and a medium range of sizes. The IVEE system will most likely implement another technique due to the few differences in the data set and the longer running times of agglomerative clustering.

4.2.7 Genetic K-means

Krishna and Narasimha Murty proposed a genetic k-means algorithm to discover clusters with a guaranteed global minimum. Genetic k-means is a combination of a genetic algorithm and a standard k-means algorithm. The genetic algorithm part of genetic k-means includes the alleles, chromosomes, populations, generations and mutations. The k-means part of genetic k-means replaces the standard crossover function with a k-means operator. Each chromosome contains a number of alleles equal to the number of data points in the data set, where each allele is equal to a number from one to k. This number associates a data point that the allele represents with a cluster centroid. Unfortunately, the genetic k-means algorithm requires the number of clusters as an input [16].

The genetic k-means algorithm initializes with a random population of

chromosomes. Throughout the algorithm, whenever a chromosome represents an impossible cluster scenario, that chromosome is fixed. Each allele in each chromosome has a chance to mutate inversely proportional of the distance of the allele's data point to the cluster centroid. The k-means operator reassigns the allele's data points to the nearest cluster centroid. Each chromosome is then tested with the fitness function–those that survive continue to the next generation. The algorithm terminates at a specified number of generations [16].

Although the genetic k-means algorithm achieves the global minimum, it appears to be much more complicated than many of the other clustering techniques. Another weakness is the requirement to know the number of clusters a priori. Due to these two main reasons and the lack of any other benefits, the IVEE system will most likely not be implementing this technique.

4.3 Neural Networks

Artificial Neural Networks (ANNs) provide a way to classify data in high dimensional spaces. Given the measurables as input, the neural network should be able to give an interest value as an output. In order for this to work, the output must be a floating point value to represent all the varying degrees of interest. Furthermore, the neural network must adapt quickly in order to handle the many observed events that can occur within a short time. The neural network must also be unsupervised as the IVEE system will not be providing input and output training pairs.

4.3.1 Hebbian Learning

Hebbian learning is an unsupervised method for training neural networks, which follows the following formula:

$$\Delta w_{ij} = \eta x_j y_i \tag{4.1}$$

It works by reinforcing the connection between pairs of neurons which fire during a training cycle [26]. This seems like an appropriate method for learning because this occurs in nature and it seems to reinforce connections the more times the neural network sees the same and similar measurables from observed events. Therefore, outputs for similar observed events should

get stronger overtime. This is the type of response required for the IVEE system. Hebbian learning also does not appear to converge to stable weights on the neural network, which is appropriate because the neural network should not have strict learning limits [26].

Some of the downsides of Hebbian learning include the networks ability to remember noise in patterns. Furthermore, unstable weights may be appropriate for the IVEE system, but unstable weights have the potential to lead to network problems. However, a rescaling technique can keep the weights in the same ratio, but within specified limits. Hebbian learning may also include a weight decay, which is similar to forgetting and a desirable response in the IVEE system. Hebbian learning handles similar input data poorly and is unable to perform a time delay between input and output [4].

4.3.2 Anti-Hebbian Learning

Anti-Hebbian learning follows the formula:

$$\Delta w_{ij} = -\eta x_j y_i \tag{4.2}$$

The only difference with Equation 4.2 from Hebbian learning is the negative sign. Hebbian learning finds the maximum variance, whereas anti-Hebbian learning finds the minimum variance. This means anti-Hebbian learning finds the decorrelation amongst data. Furthermore, anti-Hebbian learning can be used in novelty filters to ignore previously experienced input data [26].

4.3.3 Novelty Filters with Lateral Inhibition

A novelty filter simply takes the anti-Hebbian learning rule in Equation 4.2 and applies it to a processing element. In essence, a network of multiple processing elements will have two sets of data. The first set will be the data the network has already experienced and the second set will be the data the network is currently experiencing. Through this novelty filter, the previously experienced data will be ignored. To further decorrelate data, lateral inhibition can be added amongst novelty filters using anti-Hebbian learning. By placing links across neighboring neurons, winning neurons suppress the output of neighbors [26]. Novelty filters and lateral inhibition occur in nature and allow humans, as well as other animals, to ignore constant and com-

monly triggered sensory inputs. Otherwise, humans and animals would be overwhelmed by their own senses and they would find it difficult to distinguish threats in the wild.

4.3.4 Habituation

Habituation is an excellent way to determine if an observed event is novel, which is a subset of interesting observed events. Habituation follows the formula:

$$\tau \frac{dh(t)}{dt} = \alpha[h(0) - h(t)] - S(t) \tag{4.3}$$

In order to detect novel events, the most active neurons in the network are habituated. If a given input, which would be the measurables of the observed event, causes an unhabituated neuron to become active, then it can be concluded that the input is novel [19]. The interest value associated with an event might be calculated by adding the output levels of all the output layer neurons. Events that cause highly habituated neurons to fire will not contribute much to the final sum, since the habituation causes them to be suppressed.

4.3.5 Kohonen Self-Organizing Maps

Another possible solution to the problem of relating interest amongst observed events may be the Kohonen self-organizing map. These maps offer a solution through topological relations amongst the output processing elements. The output processing elements adjust their weights so that neighbors have similar weight vectors. This causes related inputs to appear near one another on the output map [26]. Thus, it should be possible to relate an interest level with distance amongst the output processing elements. Similarly, the interest level might be calculated by basing the weight difference of the input versus the winning output processing element. The problem with this would come with the initialization of the weights for the output layer. If the weights of the output layer were initialized randomly, then a near match could be found for the current input vector. This may incorrectly calculate a low interest level for a novel event. On the other hand, the initial weights of the output layer could be set to some default value.

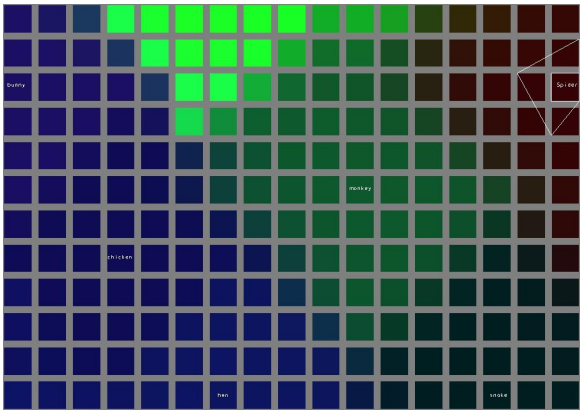

Figure 4.4: Kohonen self-organizing maps provide a method for people to visualize relations amongst high dimensional data on a two-dimensional plane [27].

The output layer may need to be more than two dimensional. In fact, the exact size of the output layer would not likely be known until several sizes have been tested and compared. The problem with picking larger output layer sizes is the amount of system memory available for the IVEE system to store observed events. Furthermore, as the size of the Kohonen map increases, the time required to traverse the map increases as well. In any case, there will always be some physical limit on the number of events that IVEE can store, recall and compare to new events.

Perhaps the size of the output layer should be limited in order to emulate a limitation of remembering in animals. Of course, the human ability to remember is much greater than that of a Kohonen self-organizing map running on an average computer system. The hard limit on the network size coupled with a forgetting scheme would enable the IVEE system to forget older events similar to many animals in nature.

Traditionally, Kohonen self-organizing maps use soft competitive learning initially, where a neighborhood of output processing elements win. Eventually, the network transitions to hard competitive learning, where only one neuron wins [26]. However, the IVEE system would continually learn and this soft-to-hard transition would either not be used or slightly modified. If the transition were not used, the IVEE system would initialize the neighborhood learning radius, where it would remain throughout the simulation. Otherwise, the IVEE system could start with an initially large neighborhood

learning radius and slowly transition to a medium sized neighborhood learning radius, where the IVEE system would remain. In any case, the IVEE system would not want to shrink the neighborhood learning radius to only a single processing element. This would cause the Kohonen self-organizing map to become stable and thus stop learning new events.

4.3.6 Adaptive Resonance Theory

The instar processing element uses the instar learning rule to update the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \eta y_i(n)(x_j(n) - w_{ij}(n)) \tag{4.4}$$

In Equation 4.4, η determines how quickly the weight converges toward the input. Furthermore, an instar processing element uses a hard limiter to output a binary result of zero or one. This output declares whether the input is similar to that of the current weights in the processing element [26].

By attaching an outstar network to the output of the instar network, called a Grossberg's Instar-Outstar network, the network can associate input and output vectors. The learning rule for Grossberg's networks is the following:

$$w_{ij}(n+1) = w_{ij}(n) + \eta x_j(n)(y_i(n) - w_{ij}(n)) \tag{4.5}$$

This simply swaps the inputs and the outputs (x and y). Grossberg modified the Instar-Outstar network to add more processing elements when the input vector did not match to any current clusters. The matching is based on the distance away from the center of the cluster. If it is within distance, the input data would update the weights of the belonging cluster. This competitive network, using the Adaptive Resonance Theory (ART), is able to cluster data and help solve the *stability-plasticity dilemma*, which is the problem of weight stability versus maintaining older associations in the networks [26].

ART appears to be promising as a clustering solution using artificial neural networks. However, the ability of the network to provide an interest value or the ability to calculate one manually based on the weights of the network is unclear.

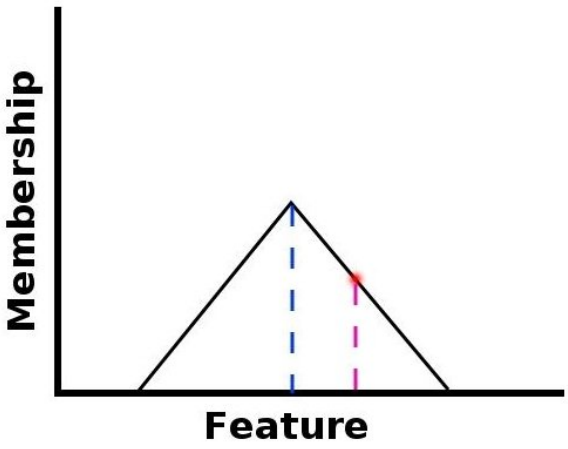

Figure 4.5: The blue dashed line represents the mean of the feature and thus the center of the triangle. The pink dashed line represents the event feature being compared with the group feature.

4.4 Group Membership

The final methodology, which was implemented for IVEE, is group membership. Group membership consists of maintaining feature vectors and associating memberships with those vectors. Group membership consists of features such as red pixel intensity, green pixel intensity, blue pixel intensity, event height, event width, location on the input video, time of event, and event velocity. An event is considered to belong to a group when it matches a certain percentage of that group. For instance, the user can define a confidence value describing the similarity required for an event to belong to a group. This similarity is calculated by inspecting each event feature as compared to the group feature.

As Figure 4.5 shows, each event feature lies somewhere on the event feature triangle, which is formed by two line segments, which meet at the coordinate defined by the feature mean and the group membership value. Event features are calculated, whether they lie above, below, or on the zero membership level mark. Therefore, if an event is drastically different from the group, it can have a negative confidence value for conforming to the group. On the other hand, all the event features may lie close to the group features except for one. This usually means that the event belongs to the group, but

the variance for the outlieing feature should be increased. IVEE will increase the variance for that feature if the event does achieve the confidence value for matching the group.

The total group confidence value is calculated by first calculating each individual feature membership with the following formula:

$$FeatureMembership = -(g_m/f_v) * |f_{\overline{x}} - e_{\overline{x}}| + g_m \tag{4.6}$$

In Equation 4.6, g_m represents the group membership level, f_v represents the group feature variance, $f_{\overline{x}}$ represents the group feature mean, and $e_{\overline{x}}$ represents the event feature mean. Equation 4.6 is calculated for each feature in the group. IVEE then averages all the feature memberships and divides this value by the group membership:

$$ConfidenceValue = \left(\left(\sum_{i=1}^{N} FeatureMembership_i \right) /N \right) /g_m \tag{4.7}$$

Equation 4.7 standardizes the results so that groups with more membership do not overpower groups with less membership. The goal is to discover the feature similarities, not allow the membership level to dictate the degree of the event belonging to the group. If the calculated confidence value from Equation 4.7 is equal to or above the user-defined confidence value, then the event belongs to the group. However, if the calculated confidence value is above zero, then the index to the group is saved in a list. If the event generates confidence values above zero but below the user-defined calculated confidence, IVEE uses the highest of these confidence values to assign an interest value to the witnessed event. This allows the interest value to reflect the relation between the event and the closest group without considering the event a part of that group. For instance, if the event matched seventy-five percent to a group, but the user-defined confidence value was set to eighty-five percent, then the event would not be considered part of the group. On the other hand, if that was the highest matched group to the event, then IVEE will compare the event to that group in order to calculate the interest value for the event. Furthermore, IVEE will create a new group for the event, since it did not pass the user-defined confidence value test, and set the initial membership of that new group to the calculated interest value of the event minus the user-defined membership value increment.

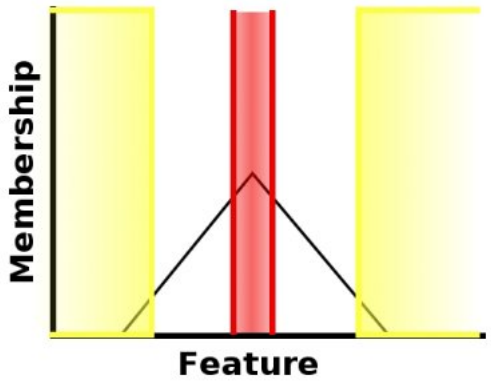

Figure 4.6: The yellow zones depict the expansion areas and the red zone depicts the contraction area.

On the other hand, if the witnessed event successfully passes the user-defined confidence value test, the event feature values will cause the group feature values to update the means and variances. The user defines the percentage weight associated with updating the group feature means. For instance, the user can define the group feature update rate to twenty percent, which causes the group feature means to move toward the event feature means by twenty percent. Similarly, the user defines the variance expansion and contraction zones along with the percentage of expansion and contraction, which is displayed in Figure 4.6. For instance, the user can define the contraction zone as the inner five percent of the feature membership triangle and the expansion zone as anything outside the inner ninety-five percent of the feature membership triangle. If a feature for the witnessed event falls within the contraction zone for the group feature, the group feature variance contracts by the user-defined percentage. If a feature for the witnessed event falls into the expansion zone, the group feature variance expands by the user-defined percentage.

Group memberships merge together if they are similar, which is defined by the user. For the first similarity assessment, a test is performed to see if one group mean has a higher membership in the other group than the first group has in itself. If this is true, then the two groups merge as shown in Figure 4.7. For the second similarity assessment, the user defines the percent confidence that the two groups are similar enough to be merged into one

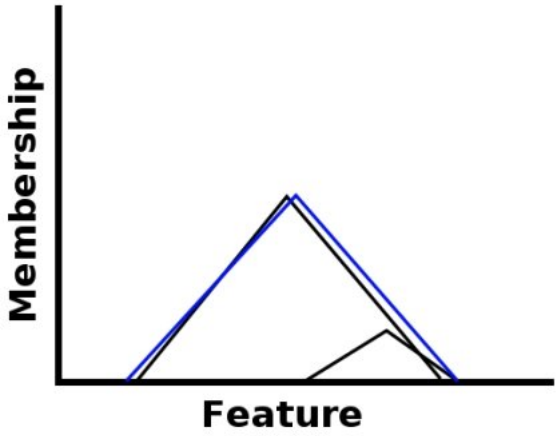

Figure 4.7: The smaller group can be merged into the larger group because the smaller group's mean value has a greater membership value in the larger group than it does in the smaller group. The blue triangle represents the final merged group.

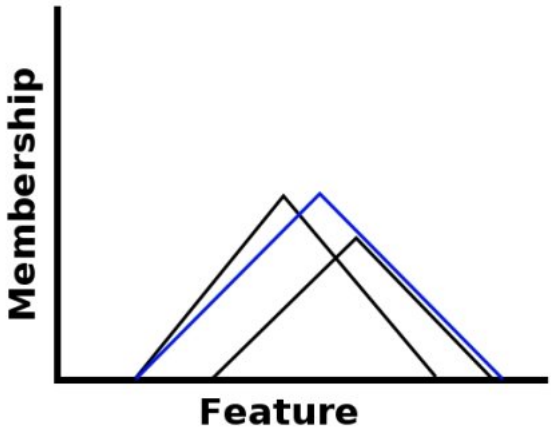

Figure 4.8: The two groups are similar enough to be merged into a single group, which is depicted by the blue triangle.

group. As Figure 4.8 shows, the resulting group is a combination of the previous groups. The user-defined percent confidence is tested by comparing how the mean of the features of one group matches within the feature means and variances of the second group, and vice-versa. Both Equation 4.6 and 4.7 are used to make this group feature comparison.

The actual merging of two groups is performed the same whether the groups were merged by similarity or one group fitting under the other. There are no user-defined constants in the group merging algorithm. The effect each original group has on the new group is based on the membership level for the original groups. Each feature of the new group is calculated by adding the features of the two original groups in ratio:

$$NewFeature = (f_g * (m_g / (m_g + m_h))) + (f_h * (m_h / (m_g + m_h))) \quad (4.8)$$

The new group must cover the same feature regions as covered by both previous groups as a minimum. This can be accomplished by calculating the absolute difference between the new group feature mean and the old group feature mean and then adding the old group feature variance onto this value:

$$NewVariance = \text{MAX}\left(\text{ABS}\left(f_n - f_g\right) + v_g, \text{ABS}\left(f_n - f_h\right) + v_h\right) \quad (4.9)$$

This is performed for both of the old groups and the maximum difference is saved as the new variance.

$$NewMembership = \text{MAX}\left(m_g, m_h\right) \quad (4.10)$$

The new membership is the maximum of the two old groups. The dates for the last update, the last forget date, and the last interest value assigned are also found by getting the max of the two old groups.

Chapter 5

Implementation

5.1 System Overview

As Figure 5.1 shows, the IVEE system consists of several modules, which work together to assign interest values to events. The Motion/Still Interest module is the entry point to the IVEE system and is the largest module. The Motion/Still Interest module calls upon the Forget Group module, which lowers group membership levels based upon the last date they were updated. The Motion/Still Interest module also calls upon the Missed Group module, which determines if an expected event did not occur. Finally, the Motion/Still Interest module calls upon the Merge Group module, which checks if groups are similar enough to be merged into a single group.

5.2 Motion/Still Interest Module

The Motion/Still Interest module declares the major data structures and calls upon all the other major modules. The Motion/Still Interest is where the main loop occurs, which reads in a single frame at a time from the input video and outputs the processed frames to disk. Several user-defined constants are found in this module, including the seconds of video that buffer interesting events in the output video, both before and after. If an interesting event occurs at 9:00 AM, lasts until 9:05 AM, and the user defines the buffer as sixty seconds, then the output video will include 8:59 AM to 9:06 AM. This feature helps people, who are watching the video output stream, to become adjusted by introducing the scene with uninteresting content and provide

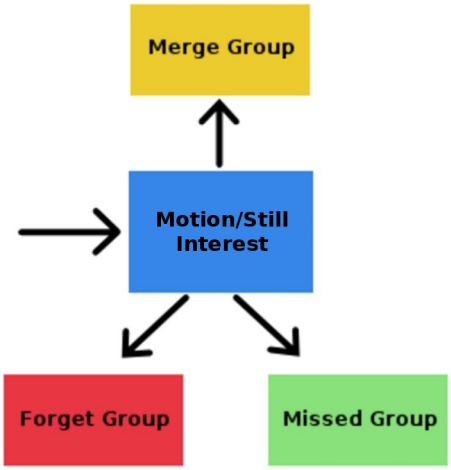

Figure 5.1: Although Matlab is not an object-oriented programming language, the source code can be broken into several software modules where each module performs a single major task.

an uninteresting conclusion. This will help people place more temporal and spatial context around the specific interesting event.

The user also defines the interest cutoff threshold for not considering events interesting anymore. A witnessed event, which has never been experienced beforehand, has an interest value of one. As this event occurs week after week, the interest value lowers and will eventually drop below the user-defined interest cutoff threshold. When the interest value is below this threshold, the event is no longer considered interesting, thus it will not have a colorful rectangle drawn about it and the event will not be written to the output video.

The IVEE system requires the user to define the initial feature variances for all new groups. The initial variances define the spread of the membership triangle formed by each feature. In general, the more the variance, the more values the feature accepts as similar to itself with respect to the user-defined confidence percentage for comparing events to groups.

The user defines the group mean update percentage, the group variance update percentage, and both the inner and outer percentage bounds for decreasing and increasing the variance. When an event matches a group, the feature means of the group move toward the feature means of the event by

the update percentage. Similarly, if a feature is inside the inner percentage zone of the feature membership triangle, the variance of that triangle is decreased. If a feature is inside the outer percentage zone of the feature membership triangle, the variance of that triangle is increased. The inner and outer percentages account for one side of the triangle membership. Therefore, if the inner percentage bound is marked as five percent for half of the triangle membership, it is ten percent for the entire triangle membership. The outer percentage technically does not stop at one hundred percent. It also covers features that lie beyond the zero membership mark, which is when the triangle side meets the zero membership line. For instance, if a group feature has a mean of one and a variance of one and the event feature has a mean of three, this would be considered to lie in an outer percentage area if set at ninety-five percent, which would cause the feature variance to widen. Of course, this would only take place if the rest of the features matched enough with the group as defined by the group confidence percentage.

Finally, the IVEE system requires that the user defines the lock time for individual groups after being updated by witnessed events. The purpose of the lock time for updated groups is to eliminate successive updates over minute periods. A typical lock time would be half a second or one second. Without this lock time, which can be set to zero seconds, groups may be updated in successive frames of the input video frame, which causes the IVEE system to become familiar with events from frame to frame. Usually, the typical behavior for the IVEE system is to become familiar with events from week to week.

5.3 Forget Group Module

The Forget Group module allows for three types of forgetting that the IVEE system can perform. The first type of forgetting is the step forget method. With this method, the IVEE system compares the date in the video against the date of each group. If the difference between the dates exceeds the user-defined forget interval, then the group membership will decrease by an amount defined by the user for each forget interval that fits into the difference. If a group membership level reaches zero or below, that group is removed from the IVEE system memory, which represents a complete loss of memory for similar events matching that group. With the step forget method, the IVEE system instantaneously drops the membership level.

The second type of forgetting is the linear forget method. This method allows the IVEE system to lose memory of an event in a linear fashion with respect to time. The user defines the slope by first defining the days to forget a standard membership amount and that amount of standard membership:

$$Slope = -StdMemUnit/Days \tag{5.1}$$

The slope is then multiplied by the absolute value of the difference between the current time and the last time the group was updated or forgotten, which is added to the old membership value. Thus, the new membership value is calculated with the following formula:

$$NewMem = Slope * \text{ABS}\,(CurTime - LastTime) + OldMem \tag{5.2}$$

The final type of forgetting is the quadratic forget method. This method allows the IVEE system to lose memory of an event in a quadratic fashion with respect to time, which at first causes IVEE to slowly lose memory of a recently updated event, but then quicker as the event is not seen again for a longer period. The user defines the same constants as the linear forget method. The slope is calculated by dividing the standard membership amount by the squared days to forget:

$$Slope = -StdMemUnit/Days^2 \tag{5.3}$$

The slope is used in the quadratic equation to calculate the reduced membership level for each group:

$$NewMem = Slope * \text{ABS}\,(CurTime - LastTime)^2 + OldMem \tag{5.4}$$

A user may easily add other forget methods by manipulating the IVEE system source code.

5.4 Missed Group Module

The Missed Group module scans through all the groups in the IVEE system's memory to discover unsatisfied events with high membership levels. The Missed Group module requires the time step from the last call to this

module, which is usually the time difference since the last processed input video stream frame. If the group has a membership level at least as high as the user-defined amount, a time feature that crosses the zero membership mark at the current time interval, and the group was not updated during the expected group time variance, then the event is considered missed. The Missed Group module writes the features and variances of the group to a text file on the disk and saves the group to another data structure, which allows for the missed expected event to be shown on the output video stream as a black and white rectangle of where the event should have taken place in the input video stream.

5.5 Merge Group Module

The Merge Group module takes as input a list of groups and performs two checks on them. If either of these two checks pass the test, then the two groups shall be merged. The first criterion is that the mean of one group has higher membership in the second group than the membership level of the first group. The second criterion is that if both group means are within a user-defined confidence percentage of the other group's mean and variance, then the two groups shall be merged. The same confidence test takes place as when deciding if an event belongs to a group in the Motion/Still Interest module.

If either of the two criteria are met, then the groups are merged together. The new merged group is affected by the means, variances, and memberships of the two groups to be merged. The equations to create the new group are performed with Equations 4.8, 4.9, and 4.10.

5.6 Major Data Structures

The data structure that remembers all the membership group information is broken down into a number of Matlab matrix files, which have the '.mat' file extension, and is referred to as the IVEE system memory. The number of files is determined by the pixels of the width of the input video stream divided by eight, and then multiplied by the height of the input video divided by eight, rounding down to the next integer. Each Matlab matrix file represents an eight by eight pixel region where events are compared against groups, which is

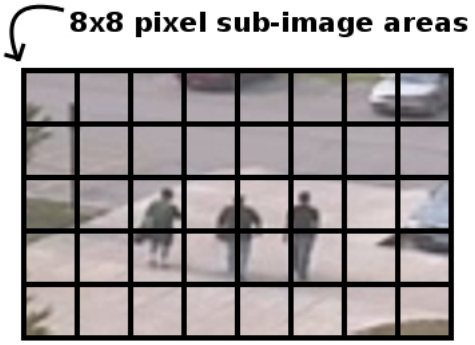

Figure 5.2: The video is dissected into eight by eight pixel sub-image regions.

depicted by Figure 5.2. For each of these regions, the groups data structure contains red pixel intensities, green pixel intensities, blue pixel intensities, heights, widths, times in hours since the beginning of Sunday, membership levels, the dates the groups were last updated, the dates the groups were last updated or forgotten, and the last interest values assigned to events by groups. The only limit to the number of groups allowed in each region is determined by the amount of RAM and hard disk space on the computer, of which RAM is usually more restricting. The user may feed the IVEE system memory back into the Motion/Still Interest module in order to retain learned knowledge from previous trials. This feature allows users to input specific times of video clips to the IVEE system throughout the week. For example, the user may only desire to input a thirty minute input video stream on Mondays from 4:30 PM to 5:00 PM without having to show the entire week. The IVEE system can learn about this time frame from week to week while ignoring the rest of the week.

Another major data structure is the interesting event structure. This structure contains a list of the interesting events that occurred for the given input video stream frame, the binary map of where the event takes place on the input video stream, the absolute time and date of the event, and the interest value of the event. This means that there could be multiple events for a single frame and that there could be no events for a single frame. A separate interesting events data structure exists for each frame on the hard disk as a Matlab matrix file.

The motion coordinates data structure is supplied by the VENUS system

Figure 5.3: The interesting value directly maps to the hue plane in HSV color space. The dashed line represents the user-defined event interest threshold.

Motion Coordinates module. This structure contains a connected components map, representing the events in motion for the frame and the number of components for the frame. The structure must be exactly as long as the number of frames in the input video. The IVEE system does not modify the motion coordinates data structure. Section 3.3.2 describes this data structure in more detail.

The missed expected events data structure allows for the drawing of the black and white rectangles in the output video stream where missed expected events took place in the input video stream. Furthermore, the data structure allows for the fading in and fading out of these rectangles, which enhance the viewing pleasure. The structure is a list containing the row, column, height, width, percent alpha value of the rectangle drawn to screen, and a flag telling if the percent alpha value is increasing or decreasing.

5.7 Memory

5.7.1 Interest Value

IVEE calculates an interest value for each motion and still event discovered in the input video stream. This interest value is a floating point decimal that ranges from 0.0, which is not interesting, to 1.0, the maximum amount of interest. The interest value directly translates into a color of a rectangle,

which is drawn about the interesting event. As Figure 5.3 shows, the interest value affects the hue plane in the HSV color space. In order to overcome sharing the same color, red, for the maximum and minimum event interest values, the user defines a threshold value where IVEE will stop drawing the rectangle. For instance, the threshold value could be 0.6. This means that the IVEE system draws all interesting objects in the range 0.6 to 1.0 only, whereas anything below 0.6 is not considered interesting.

5.7.2 Familiarization

When the IVEE system associates a witnessed event with a stored membership group, the group's membership level is increased to show the system's growing familiarity with that type of event. Membership levels start above zero and have no upper limit. As of now, the system increases a group's membership level at a constant step-wise manner. However, this functionality can be increased to include a linear, quadratic, and other membership increasing methods. The higher a group's membership level, the less interesting a similar event will be when observed by the system.

5.8 System Requirements

The IVEE was built with the following personal computer specifications:

- Microsoft Windows XP Professional SP2
- Matlab Version 7.1.0.246 (R14) Service Pack 3
- AMD Athlon 64 X2 4200+
- 2 GB RAM
- 500 GB Hard Drive
- Radeon X1600
- LAN Internet connection

5.9 Video Requirements

The IVEE system reads in Audio Video Interleave (AVI) files, which use the Microsoft MPEG-4 Video Codec V2 compression method. The IVEE system does not incorporate audio data for calculating the interest of an event. Furthermore, the IVEE system is designed for video streams captured from stationary cameras. The IVEE system has been tested using AVI input video streams with a resolution of 360 horizontal by 240 vertical pixels.

Chapter 6

Experimentation

6.1 User-Definitions

The point of the user-definition experiments are to discover the effects of altering the user set constants with a single input video stream run multiple times. Although the IVEE system will display the same general behavior of becoming less interested in the exact same events every week, the degree of interest will change based on the values of the user-defined constants.

6.1.1 Motion Constants

The motion experiments in the following sections use these constant user-defined parameters except for specific experiments as noted by each section:

```
% The seconds of video that appear before and after an interesting event.
frameBuff = 1;

% The floating point decimal between 0.0 and 1.0, inclusive.
% Events with interesting levels above or equal to this threshold
% will have a rectangle drawn about them and will be considered
% interesting.  All objects below this threshold will not be marked
% and will not be considered interesting.
interThresh = 0.75;

% Membership increase rate for new group or group update
memIncRate = 0.1;

% Initial feature variances for new group
```

```
initRedVar = 0.03;
initGreenVar = 0.03;
initBlueVar = 0.03;
initHeightVar = 3;
initWidthVar = 4;
initVelXVar = 3;
initVelYVar = 3;
initTimeVar = 0.25;

% When a group is updated, this is the percentage that the mean of the
% existing group is altered to be more like the event that matches the
% group.
gUpdate = 0.1;

% When a group is updated, this is the percentage the variance is altered,
% whether that is outward or inward.
gVarMod = 0.01;

% When a group is updated, this is the upper bound of a feature, when
% compared to the group to consider to increase the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then increase the variance.
gUpBound = 0.95;

% When a group is updated, this is the lower bound of a feature, when
% compared to the group to consider to decrease the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then decrease the variance.
gLoBound = 0.05;

% Each group becomes temporarily locked from updates after an update
% lockTime = 5.78703704e-006;  % half second in days
lockTime = 1.15740741e-005;  % one second in days

% How confident are we that this object belongs to the group? (-Inf to 1)
confidence = 0.75;

% Type of forget function: 'step', 'linear', or 'quad'
ffType = 'step';

% User-defined minimum membership level for expected events.
mcheck = 0.8;
```

Initial Feature Variances

The initial feature variances include red pixel intensity, green pixel intensity, blue pixel intensity, height, width, horizontal velocity, vertical velocity, and time of the event. By using the "People Van" video sequence, this test shows the relationship between the calculated degrees of interest and the initial feature variances. The larger the initial feature variances, the more groups accept events as being similar to themselves. In this test sequence, the initial feature variances each start at zero and increment by the values shown in the following table:

	Red	Green	Blue	Height	Width	Hor. Vel.	Vert. Vel.	Time
Init. Val.	0	0	0	0	0	0	0	0
Incr. Val.	0.05	0.05	0.05	0.5	0.5	0.5	0.5	0.05
Max Val.	1	1	1	10	10	10	10	1

Each feature variance is increased twenty times. Furthermore, for each feature variance increase, the Motion Interest module is called four times while maintaining the memory, which shows the habituation of the IVEE system if the event occurs at the same time each week. Figure A.1 depicts the results of this test, which clearly show that as the initial variances increase, the less interesting events become in the subsequent weeks because those events are matched more often to existing groups in memory.

Group Mean Update

The group mean update rate controls the rate of change of the mean for a single group to adapt to a recently matched event. The higher the update rate, the quicker the adaptation to the newest group. If this is set too high, the group quickly loses its representation for many previously matched events. However, if it is set too low, the IVEE system will take an extraordinary amount of time to learn. The user-defined group mean update value will be tested from 0.1 to 1.0 in increments of 0.1. Figure A.2 depicts the test results.

Group Variance Update

The group variance update rate determines the rate of change of the variance to cover the represented event feature spaces properly. This update rate

behaves similarly to the group mean update as it is increased or decreased. The group variance update rate was tested from 0.01 to 0.6 in increments of 0.03, which is shown in Figure A.3.

Variance Boundaries

The group variance boundaries control when the group variance update should be applied to a matched event. As Figure 4.6 shows, this constant determines the zones that cause contraction or expansion with the variances. The expansion zone value was tested from 1.0 to 0.45, whereas the contraction zone value was tested from 0.0 to 0.55, both in increments of 0.05. Figure A.4 displays the results of this test, which show little change in the degree of interest as the variance boundaries change. This most likely occurred because both boundaries were altered in the test at the same time and thus balanced the expansions and contractions of the variances.

Group Lock

The group lock value determines the period that a group cannot have its membership level altered after being matched with an event. The point of this constant is to stop the IVEE system from lowering a group's membership value in consecutive frames due to an event continuously matching a single group in that eight by eight pixel region. The group lock time was tested from 0 to 5 seconds in increments of 0.5 seconds. Figure A.5 depicts the results of this test, which clearly indicate that without the group lock constant the same event will decrease in interest across consecutive frames.

Group Confidence

The group confidence value controls the acceptance of matching events to groups based on their similarities. The greater this confidence value, the more similar the witnessed event must be to the group in order to be considered a match. The group confidence value was tested from ten percent to one hundred percent confidence in increments of ten percent. Figure A.6 displays the results of this test, which clearly show that a group confidence too small or too large causes a large amount of interest in events that should be habituated. With smaller group confidence values, events match with groups that most likely do not represent that event well, which causes the

group to update toward the newest event and thus disassociate with previously matched events. With larger group confidence values, the restrictions of matching groups to events are so high that similar events will not be matched to groups that have a good representation of the event.

6.1.2 Still Constants

The still experiments in the following sections use these constant user-defined parameters except for specific experiments as noted by each section:

```
% The seconds of video that appear before and after an interesting event.
frameBuff = 5;

% The floating point decimal between 0.0 and 1.0, inclusive.
% Events with interesting levels above or equal to this threshold
% will have a rectangle drawn about them and will be considered
% interesting.  All objects below this threshold will not be marked
% and will not be considered interesting.
interThresh = 0.75;

% Initial feature variances for new group
initRedVar = 0.03;
initGreenVar = 0.03;
initBlueVar = 0.03;
initHeightVar = 3;
initWidthVar = 4;
initTimeVar = 0.25;

% When a group is updated, this is the percentage that the mean of the
% existing group is altered to be more like the event that matches the
% group.
gUpdate = 0.1;

% When a group is updated, this is the percentage the variance is altered,
% whether that is outward or inward.
gVarMod = 0.01;

% When a group is updated, this is the upper bound of a feature, when
% compared to the group to consider to increase the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then increase the variance.
gUpBound = 0.95;

% When a group is updated, this is the lower bound of a feature, when
```

```
% compared to the group to consider to decrease the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then decrease the variance.
gLoBound = 0.05;

% Each group becomes temporarily locked from updates after an update
% lockTime = 5.78703704e-006;  % half second in days
% lockTime = 1.15740741e-005;  % one second in days
lockTime = 3.47222222e-005;  % three seconds in days

% Membership increase rate for new group or group update
memIncRate = 0.1;

% How confident are we that this object belongs to the group? (-Inf to 1)
confidence = 0.75;

% Background learning rate per frame
bgAlpha = 0.01;

% Foreground learning rate per frame
fgAlpha = 0.1;

% The threshold after differencing the bg/fg models.
% If the difference is greater than this, then the bg is considered
% different from the fg.
diffThresh = 0.3;

% Type of forget function: 'step', 'linear', or 'quad'
ffType = 'step';

% User-defined minimum membership level for expected events.
mcheck = 0.8;
```

Background Learning Rate

The background learning rate test increases the background learning rate from 0.2 percent to 2.0 percent in increments of 0.2 percent. The point of this test is to determine the change in behavior of detecting interesting events as the background learning rate is increased. The results are provided in Figure A.7, which show how the detected number of interesting events declines as the background learning rate increases. As the background learning rate becomes closer to the foreground learning rate, the difference between the two models become less and therefore less still events will be detected.

Foreground Learning Rate

The foreground learning rate test increases the foreground learning rate from five percent to fifty percent in increments of five percent. This test shows the relationship of the number of interesting events occurring versus the foreground learning rate. Figure A.8 shows how the number of interesting events increases as the foreground learning rate increases. This occurs because the difference between the foreground and background models becomes larger as the foreground model learns quicker than the background model. If the foreground learning rate is set too high, the IVEE system often detects events in motion as being events at rest. This occurs because, at the current moment, the motion coordinate data does not provide the pixels of the entire event in motion. Thus, those pixels that are not included will be counted as pixels at rest, be incorporated into the foreground model, and then be found as a difference between the foreground and background models.

Difference Threshold

The difference threshold test increases the value that determines if a still event exists by subtracting the background model from the foreground model. The difference threshold increased from 0.2 to 0.7 in increments of 0.05. As Figure A.9 shows, only the difference threshold values from 0.2 to 0.35 caused interesting events to occur. The tests involving the difference threshold values from 0.4 to 0.7 did not show any interesting events. Thus, tests six through ten were omitted from Figure A.9. In this test, the graphs do not show the whole story. Often times, the difference threshold will be determined by comparing how a person would perceive a still event against what actually was perceived as a still event by the IVEE system.

6.2 Habituation

6.2.1 Short-term

A test was developed to confirm that the IVEE system habituates repeated events over short-term intervals. Figure 6.1 shows that the IVEE system habituates events in the "People Van" video sequence when given to the system at five minute intervals. The first week is the top-left frame, the second week is the top-right frame, the third week is the bottom-left frame

Figure 6.1: The IVEE system habituates similar moving objects that repeatedly occur over a short period, as seen here in frame 266 of the "People Van" video sequence.

and the fourth week is the bottom-right frame. The IVEE system correctly habituates the events, which a person can recognize by the red, orange, and yellow rectangles, which are followed by no rectangles in the fourth week. The only error appears to be the person walking on the right-hand side of the frame in week four. However, in this scenario, the IVEE system merged groups of other people walking along this same sidewalk in later frames of the video sequence. Thus, this causes the group to adapt to more than one event at this location and that will cause the original event to be slightly interesting.

6.2.2 Long-term

The long-term experiment involves showing the "Parking Car" video sequence to the IVEE system week after week to ensure proper habituation. As Figure 6.2 shows, the IVEE system correctly habituates all events in the video sequence. The IVEE system correctly displays the colors of red, or-

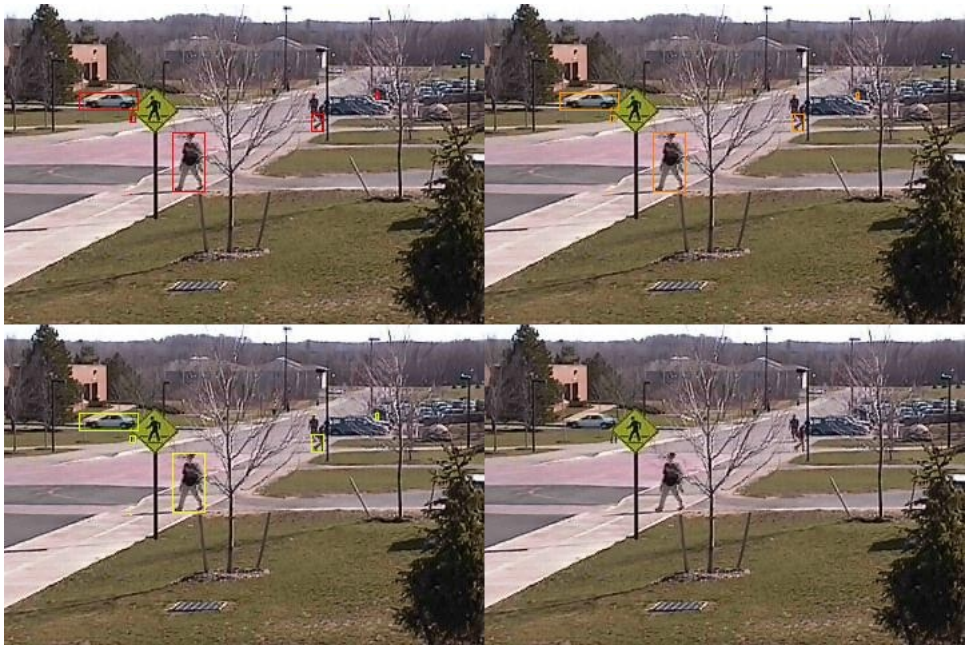

Figure 6.2: The IVEE system habituates similar moving objects that occur during the same time of the week, as seen here in frame 176 of the "Parking Car" video sequence.

ange, and yellow for the first three weeks and no rectangles for the fourth week.

6.3 Missed Expected Events

The IVEE system was tested to ensure that it properly detected missed expected events. For this test, the IVEE system was given ten weeks of the same "Crazy Guy" video sequence occurring at the same time of the week. On the eleventh week, the IVEE system was provided with the "Parking Car" video sequence, but maintaining the memory from the "Crazy Guy" video sequences. The IVEE system correctly identified the locations of the missed expected events. Figure 6.3 shows a sample of the missed expected events being drawn to screen for frame 155.

Figure 6.3: The IVEE system detects missed expected events in the video sequence, which are represented by the white and black rectangles indicating where the events should have taken place on the screen.

6.4 Accuracy

The interesting events in the "White Background" video sequence will be hand labeled as the ground truth for their degree of interestingness. The output of the IVEE system will then be compared against the ground truth to achieve a measure of accuracy. For the "White Background" video sequence, the frames where interesting motion occurs are 31 to 112, 187 to 261, 268, 269, 370 to 457, and 650 to 726. The frames where a still interesting event occurs are 458 to 649. The step-wise forget function was set for groups to lose 0.5 membership every fourteen days of inactivity. The linear and the quadratic forget functions were set for groups to lose 0.05 membership over seven days. The incremental membership rate was set at 0.1 for the step-wise forget function and 0.15 for both the linear and quadratic forget functions. The lock time for allowing groups to be updated again was one second for motion and three seconds for still.

This experiment looked at which buckets the degree of interest fell into and rated the IVEE system's accuracy based on the successful matching of the degree of interest to the ground truth. The following tables show the results of this experimentation:

Motion	Round 1	Round 2	Round 3	Round 4
Step-wise	100%	100%	100%	100%
Linear	100%	100%	100%	100%
Quad	100%	100%	100%	100%

Still	Round 1	Round 2	Round 3	Round 4
Step-wise	100%	100%	100%	100%
Linear	100%	100%	100%	100%
Quad	100%	100%	100%	100%

Chapter 7

Future Work

Although the research performed in this thesis is an excellent start to a sophisticated system, which also has many potential application uses, there are several areas that should be improved upon. There are five major areas of future work for the IVEE system. The first major area involves adding high-level, and thus abstract, modules that oversee the low-level modules. Current low-level modules simply look at the moving objects from the input video stream and associate them with a group, which determines the degree of interest in that object. However, a high-level module should track objects over their lifetime in the input video stream. By tracking an object over its lifetime, the system can gain more information about a particular scene in the video. For instance, the current IVEE system does not track objects, which can cause the degree of interest in the same object across consecutive frames to waiver (i.e. from low interest to high interest and finally to low interest again in three consecutive frames). On the other hand, a future IVEE system should realize the same object is moving across the video plane and smooth out the degrees of interest across frames. As an example, a person walking from a normal walking area to a restricted area should start as normal, then become slightly interesting, followed by a medium amount of interest, and finally extremely interesting as the person enters the restricted area. This process of assigning a degree of interest to an event over a short period is more realistic to how people assign interest.

The second major area of future work to be done is a thorough testing of various methods for assigning interest to events. Although the method created in this thesis works well, there may be a more standard approach, such as a Grow-When-Required neural network, or another method that

might work even better. Although this research included the survey of many types of methodologies, those methodologies need to be scrutinized further by implementing them, testing over a standard set of input video streams, and reporting the experiments against some ground truth measurement.

The third major area of future work to be done is to improve the performance of the IVEE system. The system should eventually become operable in real-time. The current IVEE system source code is written in Matlab. Therefore, a first area of investigation toward the realization of this goal is to port the code to another language, which would most likely be C++ with the OpenCV libraries. Furthermore, a second area of investigation should look into the maximization of efficiency for all data structures and source code. The goal of real-time operation may cause a lot of possible methodologies to be thrown away for assigning a degree of interest to an event because they are too costly to run.

The fourth major area of future work to be done is to add audio information as another dimension when analyzing the vector of input data. With the addition of sound, it would be possible to better classify abnormal activities from the norm. For instance, many abnormal and suspicious events involve higher amplitudes, like the sound of shooting a bullet from a gun, the siren from an ambulance, and the scream of someone in trouble. This information would make the difference of determining if someone laying on the grass is enjoying the sunshine or being attacked.

The fifth major area of future work to be done is to modify the representation of events. As of now, events are represented through the average color across the entire event. This could mean a person wearing a white shirt and a black pair of pants would be represented the same way as a person wearing a black shirt and a white pair of pants. This would not be appropriate because the IVEE system should be able to differentiate the two people. Instead, it may be useful to represent the event as a probability of colors in relation to the region with respect to the event. In this way, the system would notice the second person as not matching the white and black probability regions of the first person's shirt and pair of pants.

Chapter 8

Conclusions

It is unfortunate that the original VENUS system has been lost. However, the rebuilding of the VENUS system with new technologies promises to be a robust and highly applicable system. The IVEE system uses the methodology described in detail in this thesis in order to describe events in a video stream as a degree of interest. Although this research has created a system ideal for many applications, further work could create an even more robust system. The IVEE system provides accurate and precise results to detect interesting, unusual, suspect, and abnormal behavior for both events that do not normally occur and expected events that did not happen. Unlike other systems that must be fed information about the normalcy of events, the IVEE system learns for itself what is normal and what is abnormal. This allows organizations to simply plug the system in and let it do the work with minimal human interruption. These autonomous systems are the future of design and implementation, which reduce the chance of human error. The IVEE system provides great results now and shows a promising future of possibilities.

Appendices

Appendix A

Experimentation Results

The results shown in the figures are based on the following event interest color table, where the minimum values are exclusive and the maximum values are inclusive:

	Red	Orange	Yellow	Green	None
Min. Value	0.93	0.86	0.79	0.75	0.00
Max. Value	1.00	0.93	0.86	0.79	0.75

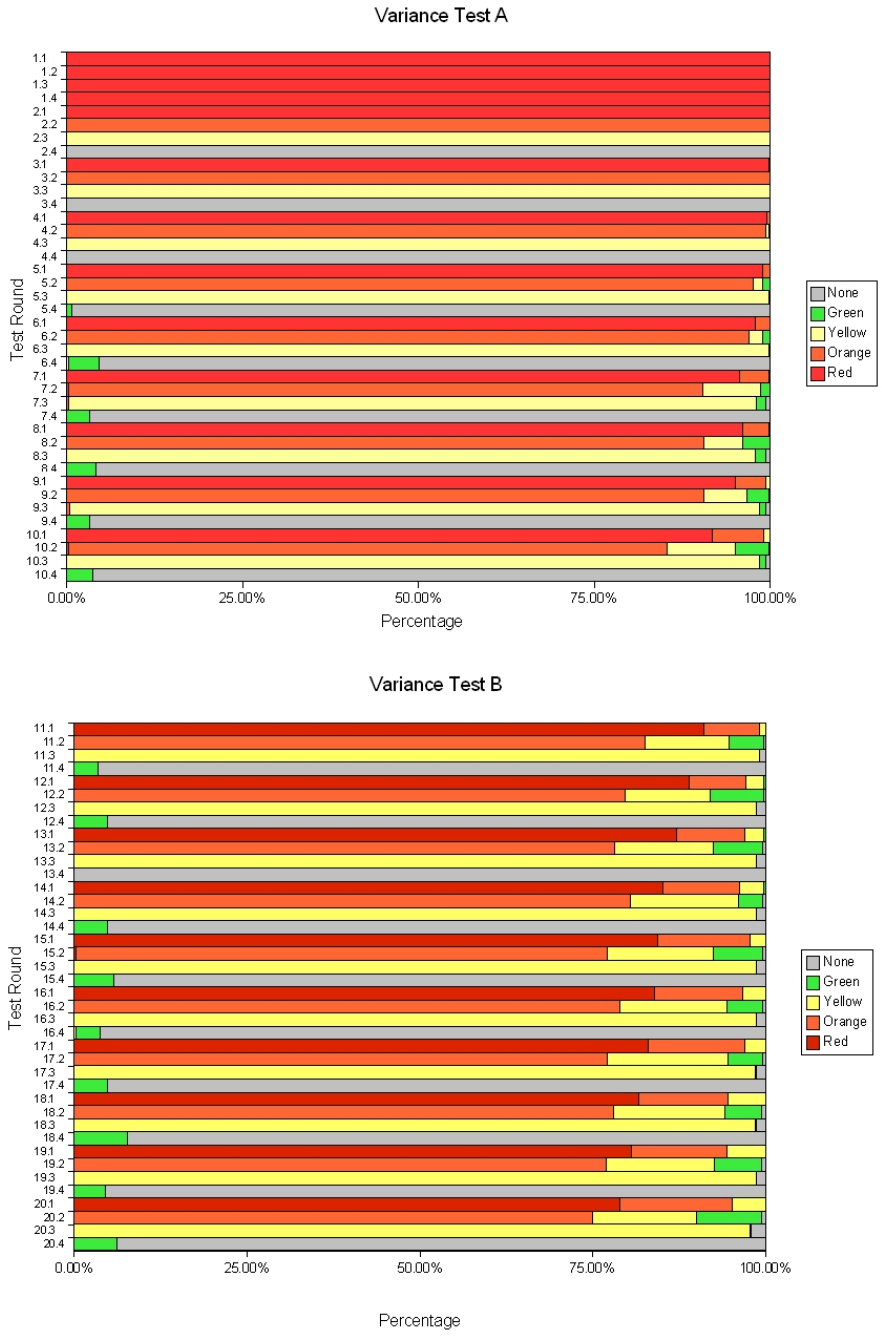

Figure A.1: The initial variance test depicts a greater acceptance of events per group as the initial variances of groups increases.

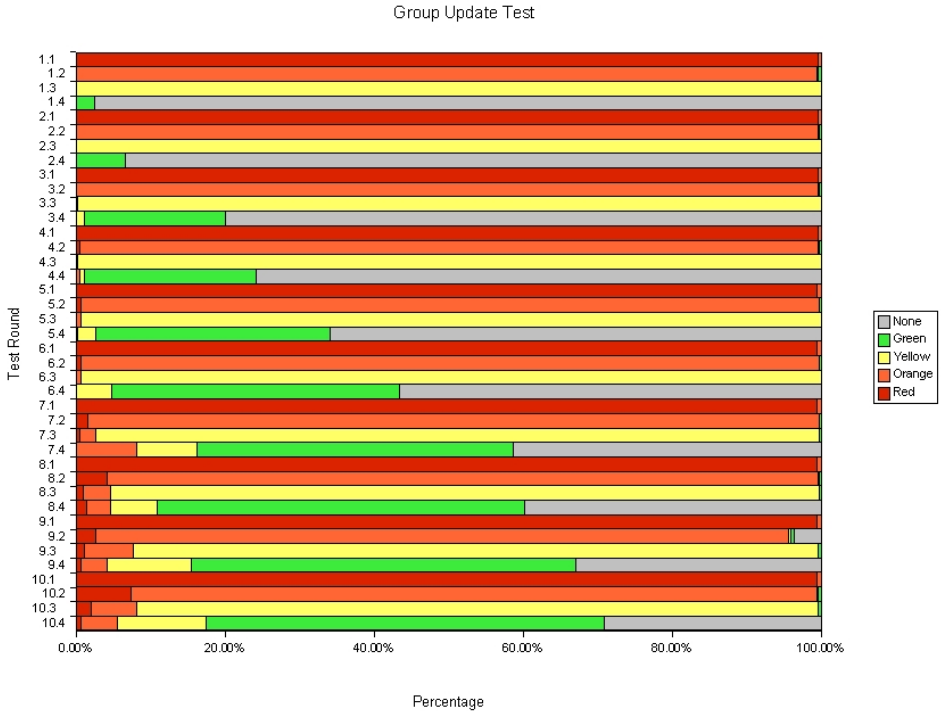

Figure A.2: The group update test shows a greater interest in the same event as the group update rate is increased. The reason for this behavior is because the group moves quickly to represent the latest matched event and thus disassociates with previously matched events.

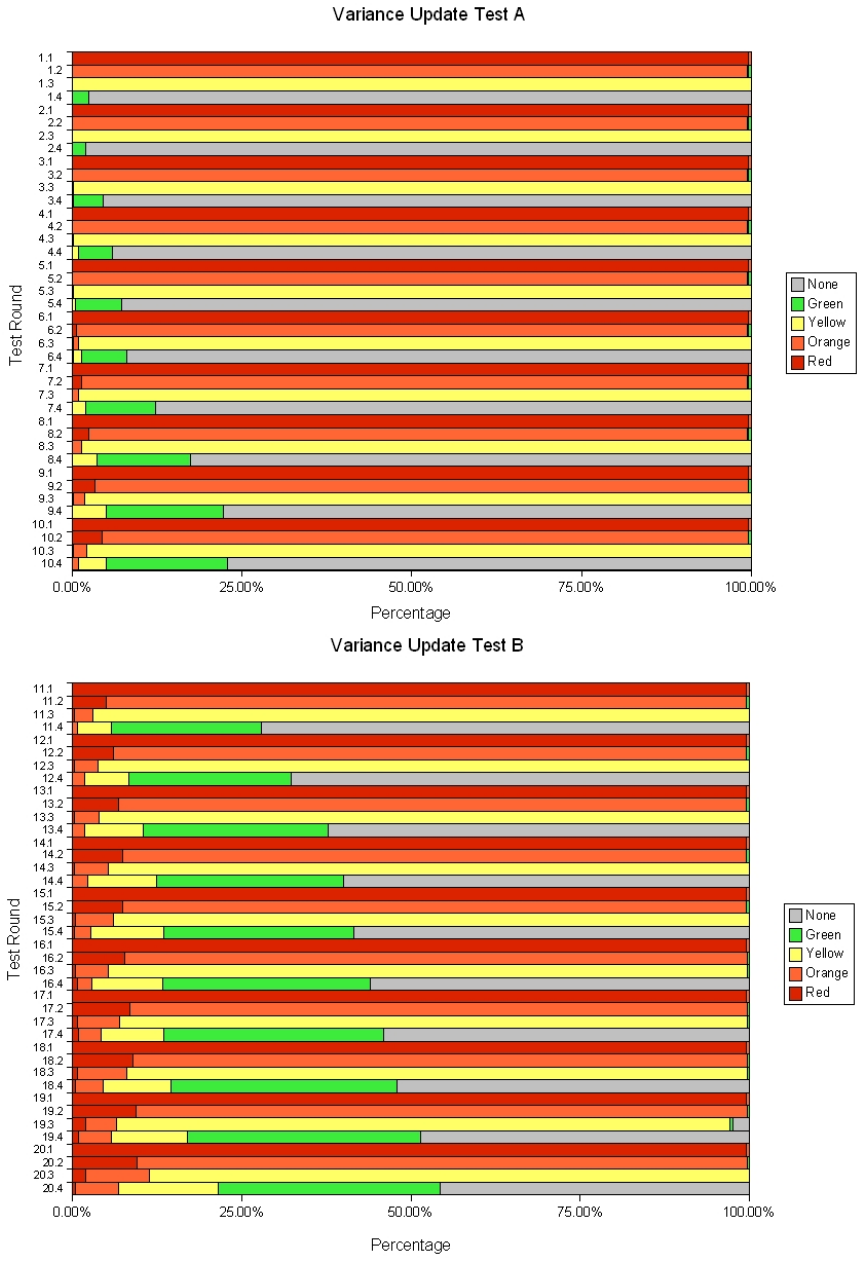

Figure A.3: The variance update test represents an extremely similar result to that of the group update test. Mainly, the IVEE system finds more interest in the same event in the fourth week because the variances adapt too quickly and thus disassociate with previously matched events.

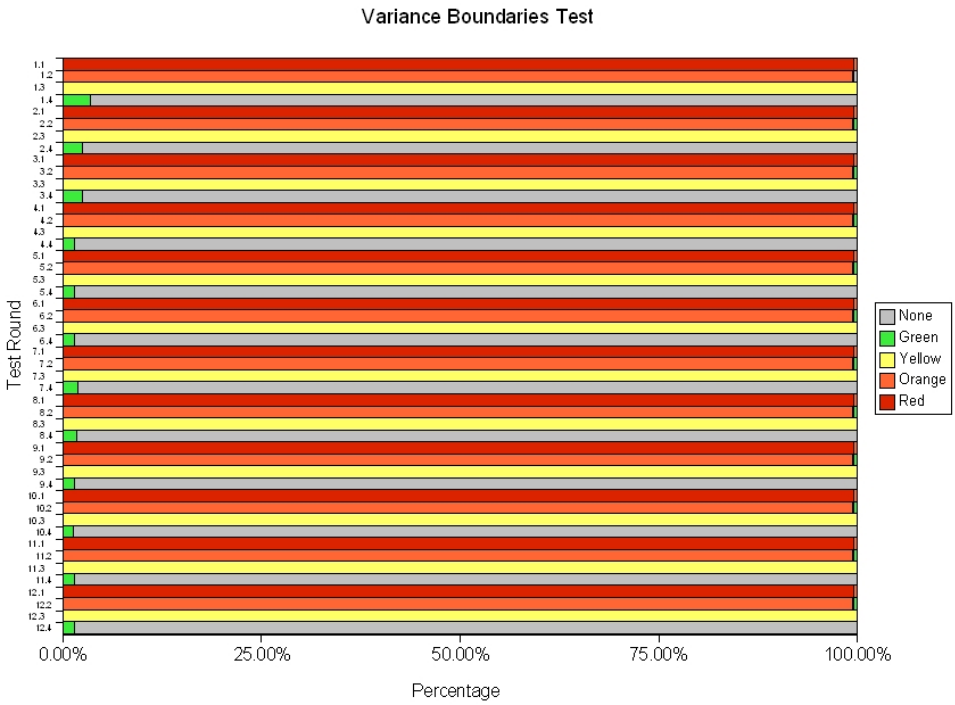

Figure A.4: The group variance boundaries test shows little change across the test rounds as the expansion zone decreased and the contraction zone increased in size.

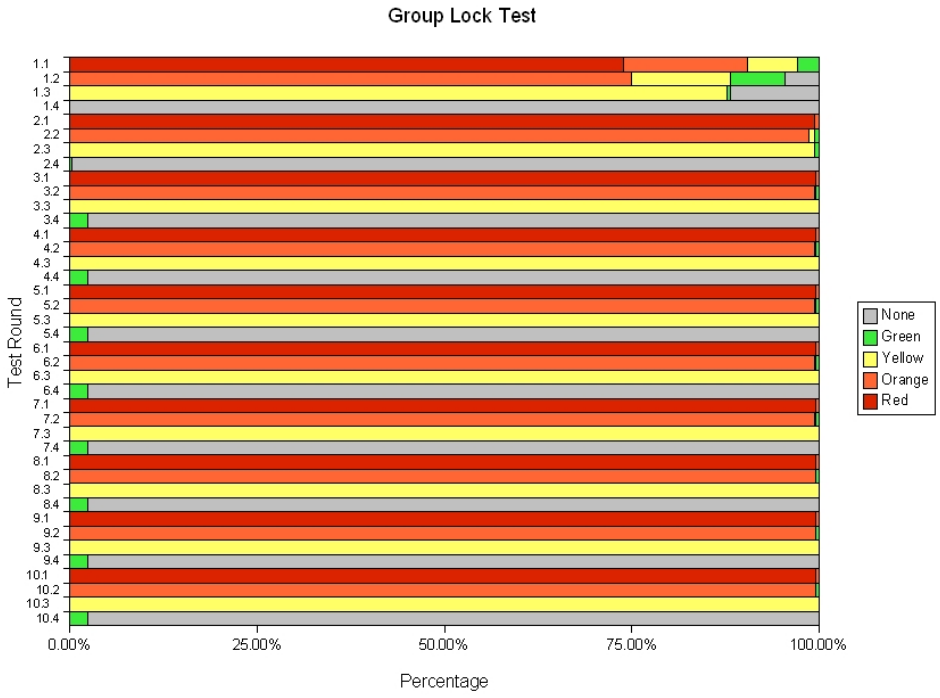

Figure A.5: The group lock test depicts how the locking of groups stops the IVEE system from losing interest in the same object over consecutive frames.

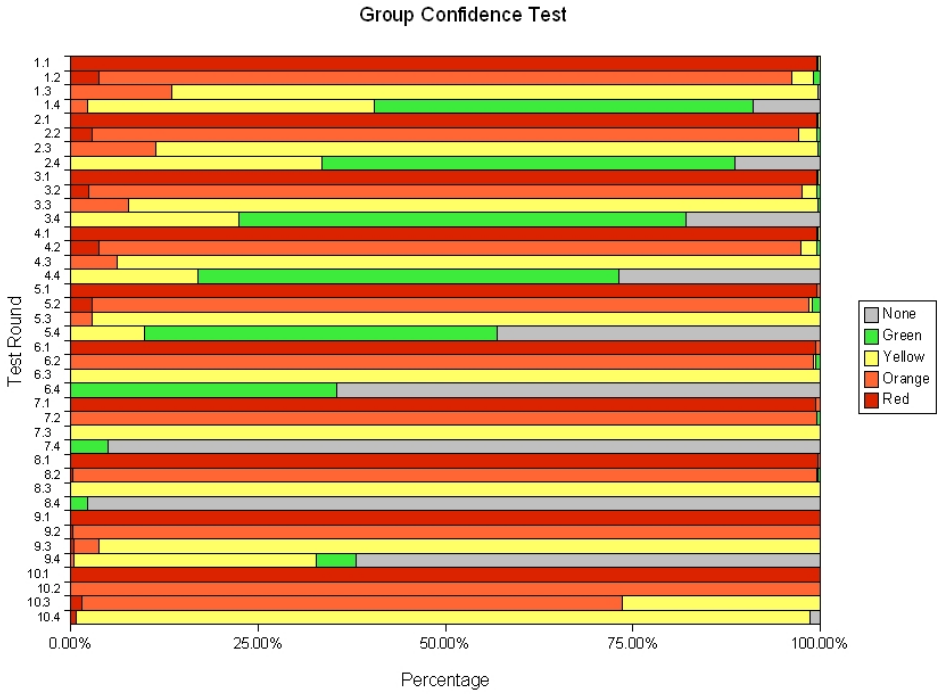

Figure A.6: The group confidence test shows how the group confidence value contains a sweet spot, where other values cause too many interesting events.

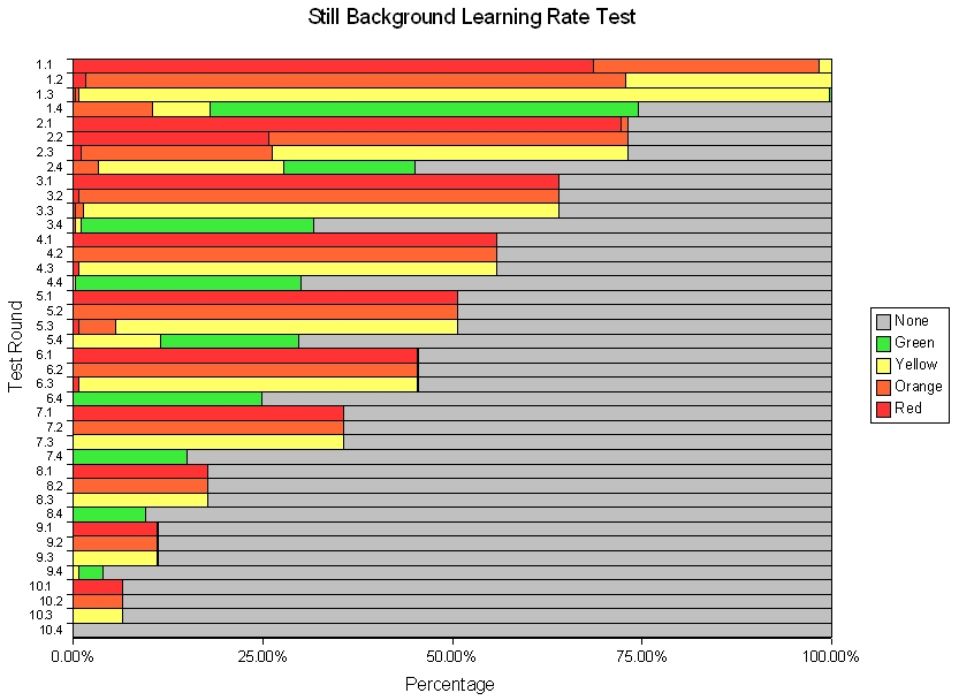

Figure A.7: The still background learning rate test results show how the number of interesting events declines with each increment of the background learning rate.

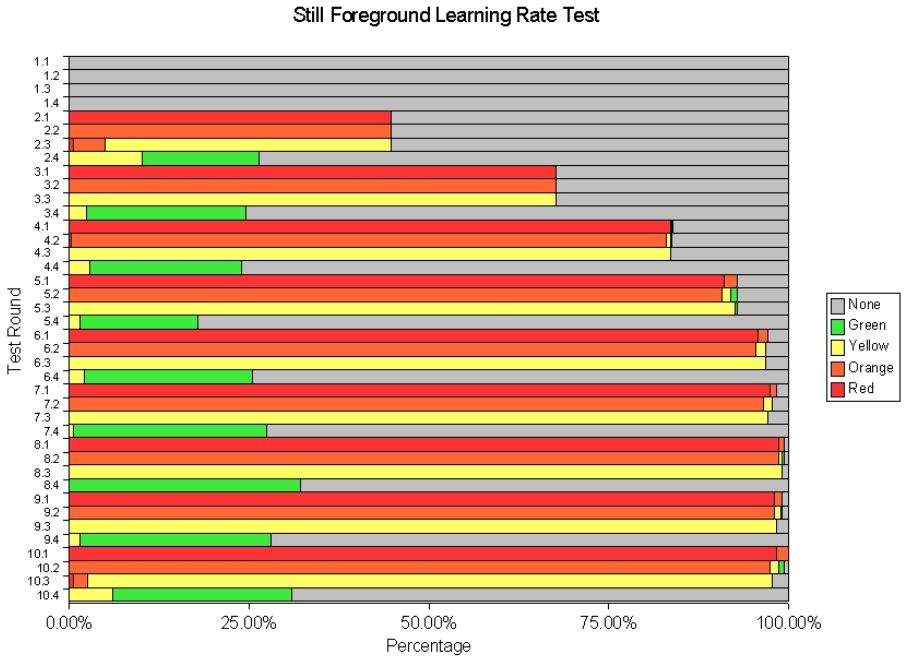

Figure A.8: The still foreground learning rate test results show how the number of interesting events increases as the foreground learning rate increases.

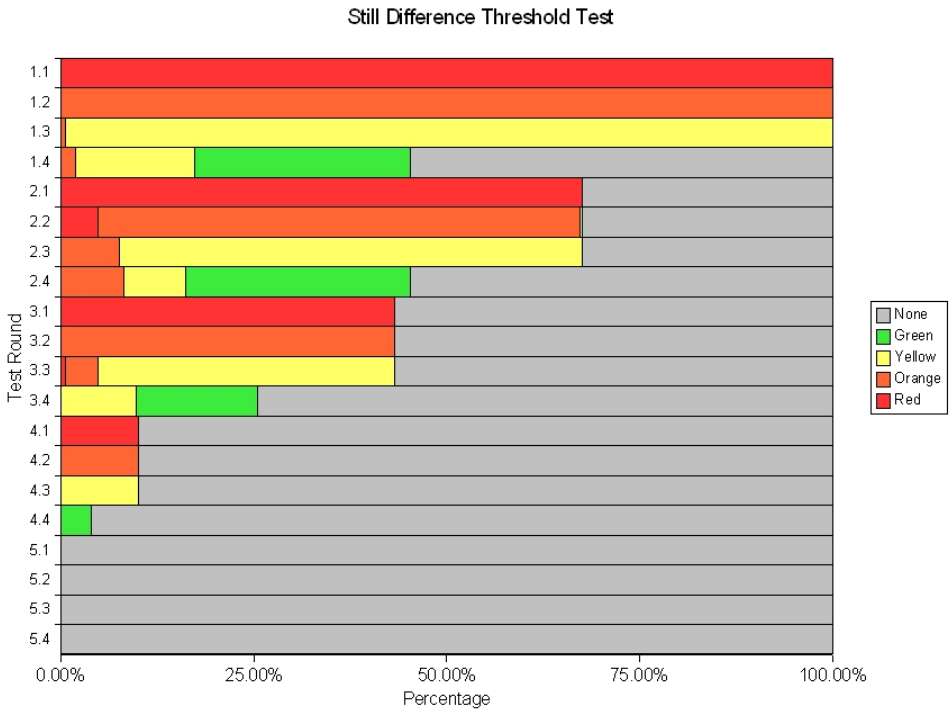

Figure A.9: The still difference threshold test results show how the system becomes insensitive to changes in the background as the threshold is increased.

Appendix B

IVEE Source Code

```
function [] = main()
%MAIN The initial entry point of the system.
%   This function shall be called first.  Please note that the back slashes
%   are used on Windows and forward slashes are used on UNIX, LINUX, and on
%   Mac OS for directories.

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

for i = 1:1

  % Set the date
  sDate = '4-Nov-2006 5:59:54';
  tDate = datenum(sDate);
  copyDir = [];
  copyDirStill = [];

  for g = 1:5

    %
    % Run Still test
    %

    stillInterest( '..\OriginalVideos\WhiteBackground.avi', ...
      'WhiteBackground', ...
      ['Results\WhiteBG\GroundTruth\Still' ...
      num2str(i) '_' num2str(g)], ...
      '..\MotionCoords\WhiteBGMotionCoords\WhiteBGMotionCoords', ...
      sDate, ...
      copyDirStill );

    % Copy the memory data from previous run
    copyDirStill = ['Results\WhiteBG\GroundTruth\Still' ...
      num2str(i) '_' num2str(g) '\data\'];

    %
    % Run Motion test
```

```
    %

    motionInterest( '..\OriginalVideos\WhiteBackground.avi', ...
      'WhiteBackground', ...
      ['Results\WhiteBG\GroundTruth\Motion' ...
      num2str(i) '_' num2str(g)], ...
      '..\MotionCoords\WhiteBGMotionCoords\WhiteBGMotionCoords', ...
      sDate, ...
      copyDir );

    % Copy the memory data from previous run
    copyDir = ['Results\WhiteBG\GroundTruth\Motion' ...
      num2str(i) '_' num2str(g) '\data\'];

    % Progress to next week
    tDate = tDate + 7;
    sDate = datestr(tDate);

  end

end

function [] = motionInterest(avifilename, writefilename, outputdir, ...
  motionCoords, datetimestring, copyDataDir)
%MOTIONINTEREST Detects motion novel and missed events in the AVI stream.
%
%  Syntax
%
%    motionInterest(avifilename, writefilename, outputdir, ...
%    motionCoords, datetimestring, copyDataDir);
%
%  Description
%
%    MOTIONINTEREST creates a series of images with rectangles about
%    events of various colors, which depict the interest level of that
%    event.  Furthermore, MOTIONINTEREST creates black and white rectangles
%    about missed expected events.
%
%  Input
%
%    avifilename - directory and filename of where the input AVI file
%    exists.  Can be a relative or absolute path.
%
%    writefilename - output naming convention of all data files in the
%    subdirectories created in the output directories.
%
%    outputdir - output directory where further subdirectories will be
%    placed for processed output image files, the data structures for each
%    region, the interesting coordinates, the missed events data,
%    and a psuedo-color representation of the motion coordinate input.
%
%    motionCoords - a string of the directory and naming convention for
%    each motion coordinate for each frame.  For instance, the string
%    '..\MotionCoords\PeopleVanMotionCoords\PeopleVanMotionCoords' would
%    represent the motion coordinates reside in the
%    '..\MotionCoords\PeopleVanMotionCoords\' directory and each motion
%    coordinate file for each frame begins with 'PeopleVanMotionCoords'.
```

```
%    This means the files will be named as follows:
%
%      PeopleVanMotionCoords_1
%      PeopleVanMotionCoords_2
%      PeopleVanMotionCoords_3
%               ...
%      PeopleVanMotionCoords_X
%
%    Where X represents the last frame of the input video stream.  There
%    must be a one to one correlation of the motion coordinates to the
%    number of frames in the video.  During the algorithm, each structure
%    is loaded for each frame.  It is expected that each loaded motion
%    coordinate structure will load the variable named 'output'.  The
%    variable 'output' has the following structure:
%
%      output.L is connected components map
%      output.num is max components for the given map
%
%    Objects across multiple frames shall be tracked by using the same
%    unique number identifier across multiple motion coordinate maps.
%    Numbers may be reused in a given input video stream, but must be
%    unique for each frame and must not be reused unless there is at least
%    one frame as a buffer between the reuse of the number.
%
%    datetimestring - A string date and time of when this video begins.
%    The rate the time increases is determined by the
%    encoded frames per second in the AVI file.  Here
%    are some examples of date/time strings:
%      '26-Jul-2006 13:46:12'
%      '30-May-1982 06:17:30'
%      '01-Dec-2072 22:00:00'
%    They are in the form dd-mmm-yyyy hh:mm:ss.
%
%    copyDataDir - The directory to copy the memory data structures for
%    each pixel region from.  The .mat files will be copied to the 'data'
%    directory, underneath the provided output directory, as specified by
%    the 'outputdir' input variable.  This value may be an empty matrix if
%    you wish to create new memory data structures for the input video
%    stream.  Each regional data structure is an array of structures, which
%    adheres to the following specifications:
%
%      r - red intensity value
%      g - green intensity value
%      b - blue intensity value
%      h - height
%      w - width
%      vx - velocity x direction
%      vy - velocity y direction
%      t - time (hours from Saturday midnight)
%      m - membership level
%      d - the date of the last update of the group
%      f - the date of the last update and forget of the group
%      i - last interest value assigned by this group to an event
%
%   Each feature contains a mean and a variance value, except for 'm', 'd',
%   'f', and 'i'.  The variance represents the distance in only one
%   direction from the mean.  Each feature can be thought of as a triangle,
```

```
%   where the top point is defined by the mean and membership level.
%   Furthermore, the side points are defined by the variance away from the
%   mean and zero membership level.  However, for comparing events against
%   groups, the triangle is allowed to extend below the zero membership
%   level.
%
%  Output
%
%    No return arguments.  All output is written to the disk at the
%    location supplied by 'outputdir'.  In the 'outputdir' directory, four
%    subdirectories will be created.  The processed image files will be
%    in the 'video' subdirectory.  The group data structures, with
%    updated values, will appear in the 'data' subdirectory.  The locations
%    of the interesting events will appear in the 'interMap' subdirectory.
%    The pseudo-colored connected components map from the 'output' variable
%    will be in the 'label' subdirectory.  Any missed events that occurred
%    during the video run will appear in 'misEvent' subdirectory in a text
%    file.
%
%  Examples
%
%    motionInterest( '..\OriginalVideos\PeopleVan.avi', ...
%      'PeopleVan', ...
%      'Results\PeopleVan\MotionX2', ...
%      '..\MotionCoords\PeopleVanMotionCoords\PeopleVanMotionCoords', ...
%      '26-Jul-2006 12:55:00', ...
%      copyDir );
%
%    motionInterest( 'OriginalVideos\PeopleVanUltimate.avi', ...
%      'TestRunTwo', ...
%      'Results\PeopleVanUltimate\Motion', ...
%      'PVUMotionCoords\PVUMC', ...
%      '9-Dec-2008 22:01:23', ...
%      [] );

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User Defined Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The seconds of video that appear before and after an interesting event.
frameBuff = 1;

% The floating point decimal between 0.0 and 1.0, inclusive.
% Events with interesting levels above or equal to this threshold
% will have a rectangle drawn about them and will be considered
% interesting.  All objects below this threshold will not be marked
% and will not be considered interesting.
interThresh = 0.75;

% Membership increase rate for new group or group update
memIncRate = 0.1;

% Initial feature variances for new group
```

```
initRedVar = 0.03;
initGreenVar = 0.03;
initBlueVar = 0.03;
initHeightVar = 3;
initWidthVar = 4;
initVelXVar = 3;
initVelYVar = 3;
initTimeVar = 0.25;

% When a group is updated, this is the percentage that the mean of the
% existing group is altered to be more like the event that matches the
% group.
gUpdate = 0.1;

% When a group is updated, this is the percentage the variance is altered,
% whether that is outward or inward.
gVarMod = 0.01;

% When a group is updated, this is the upper bound of a feature, when
% compared to the group to consider to increase the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then increase the variance.
gUpBound = 0.95;

% When a group is updated, this is the lower bound of a feature, when
% compared to the group to consider to decrease the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then decrease the variance.
gLoBound = 0.05;

% Each group becomes temporarily locked from updates after an update
% lockTime = 5.78703704e-006;  % half second in days
lockTime = 1.15740741e-005;  % one second in days

% How confident are we that this object belongs to the group? (-Inf to 1)
confidence = 0.75;

% Type of forget function: 'step', 'linear', or 'quad'
ffType = 'step';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setup Before Main Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Show the user the percent of the video data has been processed every ten
% percent.
percentDone = 0.1;

% Discover the number of frames for the AVI file
movInfo = aviinfo( avifilename );

% Save the seconds per frame
secPerFrame = 1 / movInfo.FramesPerSecond;

% Convert to days per frame because we want this fraction for adding onto
% the serial date.
% days/frame = (sec/frame) * (1m/60s) * (1h/60m) * (1d/24h)
```

```
daysPerFrame = secPerFrame * 1.1620e-005;

% Save the number of frames to buffer interesting events
frameBuffer = round( frameBuff * movInfo.FramesPerSecond );

% Find all backslashes, if none found, just start at 1
begin = strfind( avifilename, '\' ) + 1;
if isempty( begin )
  begin = 1;
end

% Save the strings that point to the output directories
videoDir = [ outputdir, '\video\' ];
labelDir = [ outputdir, '\label\' ];
dataDir = [ outputdir, '\data\' ];
interDir = [ outputdir, '\interMap\' ];
misDir = [ outputdir, '\misEvent\' ];

% Create a directory to place the images into
%   May generate a warning if the directory already exists
mkdir( videoDir );
mkdir( labelDir );
mkdir( dataDir );
mkdir( interDir );
mkdir( misDir );

% Append specific name for data files
videoDir = [videoDir writefilename];
labelDir = [labelDir writefilename];
dataDir = [dataDir writefilename];
interDir = [interDir writefilename];
misDir = [misDir writefilename];

% Create a file for missed events
fidme = fopen( [misDir '_missedEvents.txt'], 'wt' );

% Get the number of 8x8 subimages for this movie sequence
%   Any extra pixels are shaved off
numVert  = floor( movInfo.Height / 8 );
numHoriz = floor( movInfo.Width / 8 );

% Initialize motion history
motionHistory = zeros( movInfo.Height, movInfo.Width );

% The total number of elements for one plane of intensity values
totalPixels = movInfo.Height * movInfo.Width;

% If empty matrix, create new data structures
if isempty(copyDataDir)

  data = struct( 'r', [], 'g', [], 'b', [], 'h', [], 'w', [], ...
     'vx', [], 'vy', [], 't', [], 'm', [], 'd', [], 'f', [], 'i', [] );

  % Create a .mat file for each 8x8 region
  for r = 1:numVert
    for c = 1:numHoriz
      save( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );
```

```
    end
  end

else

  % Copy from the given directory to data directory
  copyfile(copyDataDir, [outputdir '\data\']);

end

% The missed events data structure is for drawing black and white
% rectangles on the output video of where an event should have taken place
% and that event's approximate size.  The structure of missed events is as
% follows:
%   r - row of missed event in 'data'
%   c - column of missed event in 'data'
%   h - height of missed event
%   w - width of missed event
%   p - the percent alpha value of the rectangle drawn to screen
%   f - the flag, when it is zero 'p' is going up, when it is one 'p' is
%   going down.
% Missed event rectangles fade in and then fade out with a rate defined by
% the user in the 'drawmisevents' function.
misEvents = struct( 'r', [], 'c', [], 'h', [], 'w', [], 'p', [], 'f', [] );

% Convert the input date string to serial format
sTime = datenum( datetimestring );

% Display when the video starts when it was recorded
disp( ['Video begins at ' datestr(sTime, 0)] );

% Maintains the uninterrupted linear video sequence with the interesting
% event.
clip = 0;

% Maintains the frame number for each clip.
sequence = 1;

% Determines what frame to stop writing images to the disk before/after
% an interesting event.
postBuff = 0;

% Begin algorithm timer
tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start reading in from the input video
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Loop for every frame of the input movie

for i = 1:movInfo.NumFrames

  % Extract the current frame
  movFrame = aviread( avifilename, i );

  % Copy the cdata and convert to double
```

```
newImage = im2double( movFrame.cdata );

% Used to show interest levels
drawImage = newImage;

% Initialize the interesting event structure,
% map - binary map, where zero represents no motion and one represents
%       motion of the object
% time - serial time/date of event
% interest - the interesting value from 0 to 1, inclusive.  One is the
%            highest level of interest.
interMap = struct('map', [], 'time', [], 'interest', []);

% Load this frame's motion coordinates into variable 'output'
load([motionCoords '_' num2str(i)]);

% For now, assume no interesting objects
drewObject = 0;

% Extract serial date to vector date
vTime = datevec( sTime );

% Get the hours since Saturday midnight (beg. of Sunday)
hrs = (((weekday(datestr(sTime,1))-1)*24) + vTime(4) +...
  (vTime(5)/60) + (vTime(6)/3600));

% Get the unique numbers of the objects in motion
objNums = unique(output.L)';
objNums = objNums(2:end);

% Loop for each object in motion
for j = objNums

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Record observables from object in motion
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Save the current object in motion
  object = output.L == j;

  % Get the indices of the current object
  coords = find( object );

  % Get the rows/cols of object
  [row,col] = find( object );

  % Calculate the center of the object (8x8 pixel regions)
  r = ceil( floor( ( max( row ) + min( row ) ) / 2 ) / 8 );
  c = ceil( floor( ( max( col ) + min( col ) ) / 2 ) / 8 );

  % Subtract center of object now and previous frame to achieve velocity
  hobject = motionHistory == j;
  [hrow,hcol] = find( hobject );

  % Do not calculate velocity if no history of object
  if ~isempty(hrow) && ~isempty(hcol)
    vely = round( ((max(row) + min(row)) / 2) ...
```

```
    - ((max(hrow) + min(hrow)) / 2) );
  velx = round( ((max(col) + min(col)) / 2) ...
    - ((max(hcol) + min(hcol)) / 2) );
else
  vely = 0;
  velx = 0;
end

% Calculate the mean of the intensities where the motion was found.
red = mean( newImage( coords ) );
green = mean( newImage( coords + totalPixels ) );
blue = mean( newImage( coords + 2 * totalPixels ) );

% Obtain the width/height of object in motion
width = max( col ) - min( col ) + 1;
height = max( row ) - min( row ) + 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load the appropriate data region into 'data'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Store observable data into appropriate region
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check to see if observed data matches previous data
match = 0;
besttest = 0;  % Only consider groups above this value
bestmember = 0;
ind = [];
bdta = 0;
bdtb = 0;
bdtc = 0;
x = 1;
b = [];
bestgroup = 0;

% Loop through all the groups
while x <= size(data.r,1)

  % Forget groups as necessary
  switch lower(ffType)
    case {'step'}
      [data,del] = fgStep( data, x, sTime );
    case {'linear'}
      [data,del] = fgLinear( data, x, sTime );
    case {'quad'}
      [data,del] = fgQuadratic( data, x, sTime );
    otherwise
      disp( 'Unkown forget type.  Defaulting to step.' );
      [data,del] = fgStep( data, x, sTime );
  end

  % If deleted/forgotten, continue searching
  if del == 1
```

```
  continue;
end

% If the data is within proper variance
dr = abs( data.r(x,1) - red );
dg = abs( data.g(x,1) - green );
db = abs( data.b(x,1) - blue );
dh = abs( data.h(x,1) - height );
dw = abs( data.w(x,1) - width );
dvx = abs( data.vx(x,1) - velx );
dvy = abs( data.vy(x,1) - vely );
dta = abs( data.t(x,1) - hrs );

% Check for Sat to Sun rollover
% one week = 168 hours
dtb = data.t(x,1) - (hrs - 168);
dtc = (hrs + 168) - data.t(x,1);

% Get the minimum time as a check for both the Sat to Sun rollover
% and for normal time operations during the week
dtd = min( dtb, dtc );
dt = min( dta, dtd );

% Get membership for each observable [y=mx+b]
mr = -(data.m(x)/data.r(x,2))*dr + data.m(x);
mg = -(data.m(x)/data.g(x,2))*dg + data.m(x);
mb = -(data.m(x)/data.b(x,2))*db + data.m(x);
mh = -(data.m(x)/data.h(x,2))*dh + data.m(x);
mw = -(data.m(x)/data.w(x,2))*dw + data.m(x);
mvx = -(data.m(x)/data.vx(x,2))*dvx + data.m(x);
mvy = -(data.m(x)/data.vy(x,2))*dvy + data.m(x);
mt = -(data.m(x)/data.t(x,2))*dt + data.m(x);

% Combine observable membership.
% This is the membership level of the observable data in relation
% to the group.
member = mean( [mr mg mb mh mw mvx mvy mt], 2 );

% Divide by max group membership to standardize results
test = member / data.m(x);

% Is this the best match so far?
if test >= besttest

  % We found at least one match
  match = 1;

  % Maintain list of indicies
  ind(end+1) = x;

  % Test that confidence
  if test >= confidence

    % This event passes the confidence level for this group.  Save
    % the best group information to be updated with this event later.

    % Save best info for later
```

```
      bestmember = member;
      b = x;
      besttest = test;
      bdta = dta;
      bdtb = dtb;
      bdtc = dtc;

      % Don't bother looking for groups to get a relative interest
      % level, but not to update.  We have one that passed the
      % confidence level, so we will at least update this group.
      bestgroup = 1;

    elseif bestgroup == 0

      % We haven't found a group that has passed the confidence level,
      % but save this group because it is the best we have seen thus
      % far.  We will not update this group, but simply use it to
      % compare to the current event, which will let us define an
      % interest level for the event and a membership level for the new
      % group.

      % Save best info for later
      bestmember = member;
      b = x;
      besttest = test;
      bdta = dta;
      bdtb = dtb;
      bdtc = dtc;

    end

  end

  % Increment x
  x = x + 1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine if a match was found
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if match == 1 && bestgroup == 1

  % We found a similar group to update...

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Modify existing group
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   % Alter the center data point according to weight
  data.r(b,1) = (1-gUpdate)*data.r(b,1) + gUpdate*red;
  data.g(b,1) = (1-gUpdate)*data.g(b,1) + gUpdate*green;
  data.b(b,1) = (1-gUpdate)*data.b(b,1) + gUpdate*blue;
  data.h(b,1) = (1-gUpdate)*data.h(b,1) + gUpdate*height;
  data.w(b,1) = (1-gUpdate)*data.w(b,1) + gUpdate*width;
  data.vx(b,1) = (1-gUpdate)*data.vx(b,1) + gUpdate*velx;
```

```
data.vy(b,1) = (1-gUpdate)*data.vy(b,1) + gUpdate*vely;

% Select the smallest time difference
if bdtb <= bdta && bdtb <= bdtc
  % The hours must be moved to next week
  thrs = hrs - 168;
elseif bdtc <= bdta && bdtc <= bdtb
  % The hours must be moved to prev week
  thrs = hrs + 168;
else
  % Same week
  thrs = hrs;
end

data.t(b,1) = (1-gUpdate)*data.t(b,1) + gUpdate*thrs;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modify variance if new observation in extreme middle or
% extreme outside of group.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (abs( red - data.r(b,1) ) / data.r(b,2)) > gUpBound
  % Increase variance
  data.r(b,2) = (1+gVarMod)*data.r(b,2);
elseif (abs( red - data.r(b,1) ) / data.r(b,2)) ...
    < gLoBound
  % Decrease variance
  data.r(b,2) = (1-gVarMod)*data.r(b,2);
end

if (abs( green - data.g(b,1) ) / data.g(b,2)) > gUpBound
  % Increase variance
  data.g(b,2) = (1+gVarMod)*data.g(b,2);
elseif (abs( green - data.g(b,1) ) / data.g(b,2)) ...
    < gLoBound
  % Decrease variance
  data.g(b,2) = (1-gVarMod)*data.g(b,2);
end

if (abs( blue - data.b(b,1) ) / data.b(b,2)) > gUpBound
  % Increase variance
  data.b(b,2) = (1+gVarMod)*data.b(b,2);
elseif (abs( blue - data.b(b,1) ) / data.b(b,2)) ...
    < gLoBound
  % Decrease variance
  data.b(b,2) = (1-gVarMod)*data.b(b,2);
end

if (abs( height - data.h(b,1) ) / data.h(b,2)) > gUpBound
  % Increase variance
  data.h(b,2) = (1+gVarMod)*data.h(b,2);
elseif (abs( height - data.h(b,1) ) / data.h(b,2)) ...
    < gLoBound
  % Decrease variance
  data.h(b,2) = (1-gVarMod)*data.h(b,2);
end
```

```
if (abs( width - data.w(b,1) ) / data.w(b,2)) > gUpBound
  % Increase variance
  data.w(b,2) = (1+gVarMod)*data.w(b,2);
elseif (abs( width - data.w(b,1) ) / data.w(b,2)) ...
    < gLoBound
  % Decrease variance
  data.w(b,2) = (1-gVarMod)*data.w(b,2);
end

if (abs( velx - data.vx(b,1) ) / data.vx(b,2)) > gUpBound
  % Increase variance
  data.vx(b,2) = (1+gVarMod)*data.vx(b,2);
elseif (abs( velx - data.vx(b,1) ) / data.vx(b,2)) ...
    < gLoBound
  % Decrease variance
  data.vx(b,2) = (1-gVarMod)*data.vx(b,2);
end

if (abs( vely - data.vy(b,1) ) / data.vy(b,2)) > gUpBound
  % Increase variance
  data.vy(b,2) = (1+gVarMod)*data.vy(b,2);
elseif (abs( vely - data.vy(b,1) ) / data.vy(b,2)) ...
    < gLoBound
  % Decrease variance
  data.vy(b,2) = (1-gVarMod)*data.vy(b,2);
end

if (abs( thrs - data.t(b,1) ) / data.t(b,2)) > gUpBound
  % Increase variance
  data.t(b,2) = (1+gVarMod)*data.t(b,2);
elseif (abs( thrs - data.t(b,1) ) / data.t(b,2)) ...
    < gLoBound
  % Decrease variance
  data.t(b,2) = (1-gVarMod)*data.t(b,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ensure proper hours per week
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Fix the mean time if a rollover occurs across Sat and Sun
if data.t(b,1) < 0
  % Move to prev week
  data.t(b,1) = data.t(b,1) + 168;
elseif data.t(b,1) > 168
  % Move to next week
  data.t(b,1) = data.t(b,1) - 168;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine if group is locked
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Is this group locked temporarily?
if data.d(b) + lockTime < sTime

  % If not locked...
```

```
    % Increase the membership (height)
    data.m(b) = data.m(b) + memIncRate;

    % Remember last assigned membership
    data.i(b) = bestmember;

    % Set last update date/time (for forgetting and lockout)
    data.d(b) = sTime;
    data.f(b) = sTime;

  else

    % If locked...

    % Use last assigned color for this event
    bestmember = data.i(b);

  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Merge similar groups
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  data = checkgroups(data,ind,'motion');

elseif match == 1 && bestgroup == 0

  % We found the most similar group, but it does not pass the
  % confidence level.  We will create a new group, but the membership
  % will be based off the related interest level of the event to this
  % group we found.

  % Place group at end of array
  z = size(data.r,1)+1;

  % Set the group mean value
  data.r(z,1) = red;
  data.g(z,1) = green;
  data.b(z,1) = blue;
  data.h(z,1) = height;
  data.w(z,1) = width;
  data.vx(z,1) = velx;
  data.vy(z,1) = vely;
  data.t(z,1) = hrs;

  % Set the initial variance
  data.r(z,2) = initRedVar;
  data.g(z,2) = initGreenVar;
  data.b(z,2) = initBlueVar;
  data.h(z,2) = initHeightVar;
  data.w(z,2) = initWidthVar;
  data.vx(z,2) = initVelXVar;
  data.vy(z,2) = initVelYVar;
  data.t(z,2) = initTimeVar;

  % Set initial membership to max of init membership or bestmember
```

```
  bestmember = max( bestmember, memIncRate );
  data.m(z) = bestmember;

  % Set last update date/time (for forgetting and lockout)
  data.d(z) = sTime;
  data.f(z) = sTime;

  % Remember last assigned membership.
  % But subtract the membership increase rate because this is the
  % reported interest level.
  bestmember = bestmember - memIncRate;
  data.i(z) = bestmember;

else

  % No groups similar enough to this event...

  %%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Create a new group
  %%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Place group at end of array
  b = size(data.r,1)+1;

  % Set the group mean value
  data.r(b,1) = red;
  data.g(b,1) = green;
  data.b(b,1) = blue;
  data.h(b,1) = height;
  data.w(b,1) = width;
  data.vx(b,1) = velx;
  data.vy(b,1) = vely;
  data.t(b,1) = hrs;

  % Set the initial variance
  data.r(b,2) = initRedVar;
  data.g(b,2) = initGreenVar;
  data.b(b,2) = initBlueVar;
  data.h(b,2) = initHeightVar;
  data.w(b,2) = initWidthVar;
  data.vx(b,2) = initVelXVar;
  data.vy(b,2) = initVelYVar;
  data.t(b,2) = initTimeVar;

  % Set initial membership
  data.m(b) = memIncRate;
  bestmember = 0;

  % Set last update date/time (for forgetting and lockout)
  data.d(b) = sTime;
  data.f(b) = sTime;

  % Remember last assigned membership
  data.i(b) = bestmember;

end
```

```
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Draw the interest color of the event
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Draw the rectangle if above interesting threshold
  if (1-bestmember) >= interThresh

    % Draw a box around the object according to membership
    drawImage = drawObject( drawImage, row, col, bestmember );

    % Set the flag that we drew an object
    drewObject = 1;

    % Save to the interesting event map
    index = numel( interMap.time ) + 1;
    %interMap.frame = i;
    interMap(index).map = object;
    interMap(index).time = sTime;
    interMap(index).interest = (1-bestmember);

  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Save the data structure to disk
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  save( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );

end  % Loop to the next event in the scene...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Record and draw missed expected events
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Write missed events to text file and return 'pointers' of missed
% events in the data structure
misEvents = missedgroups(dataDir,numVert,numHoriz,daysPerFrame,...
  sTime,hrs,fidme,misEvents);

% Draw missed events
if ~isempty(misEvents.r)
  [drawImage,misEvents] = drawmisevents(drawImage,movInfo,misEvents);

  % Write this image to disk
  drewObject = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Write images to disk for the current frame and buffering frames
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If we drew a rectangle indicating an interesting event
if i <= postBuff || drewObject == 1

  % If the current frame is beyond the post buffer
  if i > postBuff
```

```
    % Write previous input movie frames
    preBuff = i - frameBuffer;

    % Check for beginning of movie
    if preBuff < 1
      preBuff = 1;
    end

    % Continue the last clip because the buffers overlap
    if preBuff < postBuff
      preBuff = postBuff + 1;
    else
      % Go to next clip, reset sequence
      clip = clip + 1;
      sequence = 1;
    end

    for x = preBuff:(i-1)
      % Extract the frame
      tmpFrame = aviread( avifilename, x );

      % Copy the cdata and convert to double
      tmpImage = im2double( tmpFrame.cdata );

      % Write the file
      imwrite( tmpImage, [videoDir '.' num2str( clip ) '.' ...
        num2str( sequence ) '.jpg'] );

      sequence = sequence + 1;
    end

  end

  % Write this interesting event to the disk
  imwrite( drawImage, [videoDir '.' num2str( clip ) '.' ...
    num2str( sequence ) '.jpg'] );

  sequence = sequence + 1;

  % If interesting objects in this frame
  if drewObject == 1
    % Save the frame where the Post Buffer would end
    postBuff = i + frameBuffer;
  end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of the main loop cleanup
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Save the interesting events to disk for this frame
save( [interDir '_interMap_' num2str(i)], 'interMap' );

% Write the psuedo-color motion coords to disk
imwrite( label2rgb( output.L ), [ labelDir ...
  , '.', num2str( i ), '.jpg' ] );
```

```
  % Save the current motion into the history
  motionHistory = output.L;

  % Increment the time with days per frame
  sTime = sTime + daysPerFrame;

  % Progress report every ten percent
  if abs( ( i / movInfo.NumFrames ) - percentDone ) < 0.01
    disp( [ num2str( percentDone * 100 ), '% complete.' ] );
    percentDone = percentDone + 0.1;
  end

end % Loop to the next input video frame...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cleanup after main loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Last check to forget groups
forgetgroups(dataDir,numVert,numHoriz,sTime,ffType);

% Say when the video time ends when the video was recorded
disp( ['Video ends at ' datestr(sTime, 0)] );

% Close writing to the missed events text file
status = fclose(fidme);

% Report if error
if status == -1
  disp( 'Unable to close the missed events file.' );
end

% Print out algorithm time
n = toc;
disp( [ num2str( n / 60 ), ' minutes to complete algorithm.' ] );



function [ drawImage ] = drawObject( drawImage, row, col, hsvColor )
% DRAWOBJECT Draws a rectangle at the given coordinates.

% Save the beginning/ending rows/columns for the box
upLeftX = min( col );
upLeftY = min( row );
botRightX = max( col );
botRightY = max( row );

% Convert HSV color to RGB
rgbColor = hsv2rgb( cat( 3, hsvColor, 1.0, 1.0 ) );

%%%%%%%%%%%%%%%%%%%%%%%%
% Draw the bounding box
%%%%%%%%%%%%%%%%%%%%%%%%

% North
drawImage( upLeftY, upLeftX:botRightX, 1 ) = ...
```

```
  rgbColor( 1, 1, 1 );
drawImage( upLeftY, upLeftX:botRightX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( upLeftY, upLeftX:botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% South
drawImage( botRightY, upLeftX:botRightX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( botRightY, upLeftX:botRightX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( botRightY, upLeftX:botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% East
drawImage( upLeftY:botRightY, botRightX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( upLeftY:botRightY, botRightX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( upLeftY:botRightY, botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% West
drawImage( upLeftY:botRightY, upLeftX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( upLeftY:botRightY, upLeftX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( upLeftY:botRightY, upLeftX, 3 ) = ...
  rgbColor( 1, 1, 3 );



function [drawImage,me] = drawmisevents(drawImage,movInfo,me)
% DRAWMISEVENTS Draws black & white rectangles around missed events.

% User-defined fade in/out of missed events
alpha = 0.1;

% Loop for every missed event
x = 1;
while x <= numel(me.r)

  % Save the beginning/ending rows/columns for the box
  upLeftX = (me.c(x)*8-4) - ceil(me.w(x)/2);
  upLeftY = (me.r(x)*8-4) - ceil(me.h(x)/2);
  botRightX = (me.c(x)*8-4) + ceil(me.w(x)/2);
  botRightY = (me.r(x)*8-4) + ceil(me.h(x)/2);

  % Limit the boundaries to the output video
  if upLeftX < 1
    upLeftX = 1;
  end
  if upLeftY < 1
    upLeftY = 1;
  end
  if botRightX > movInfo.Width
    botRightX = movInfo.Width;
```

```
end
if botRightY > movInfo.Height
  botRightY = movInfo.Height;
end

% RGB colors
white = [1 1 1];
black = [0 0 0];

%%%%%%%%%%%%%%%%%%%%%%%%
% Draw the bounding box
%%%%%%%%%%%%%%%%%%%%%%%%

% North
drawImage( upLeftY, upLeftX:botRightX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 1 );
drawImage( upLeftY, upLeftX:botRightX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 2 );
drawImage( upLeftY, upLeftX:botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 3 );

drawImage( upLeftY+1, upLeftX+1:botRightX-1, 1 ) = me.p(x)*black(1) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 1 );
drawImage( upLeftY+1, upLeftX+1:botRightX-1, 2 ) = me.p(x)*black(2) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 2 );
drawImage( upLeftY+1, upLeftX+1:botRightX-1, 3 ) = me.p(x)*black(3) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 3 );

% South
drawImage( botRightY, upLeftX:botRightX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 1 );
drawImage( botRightY, upLeftX:botRightX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 2 );
drawImage( botRightY, upLeftX:botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 3 );

drawImage( botRightY-1, upLeftX+1:botRightX-1, 1 ) = me.p(x)*black(1)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 1 );
drawImage( botRightY-1, upLeftX+1:botRightX-1, 2 ) = me.p(x)*black(2)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 2 );
drawImage( botRightY-1, upLeftX+1:botRightX-1, 3 ) = me.p(x)*black(3)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 3 );

% East
drawImage( upLeftY:botRightY, botRightX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 1 );
drawImage( upLeftY:botRightY, botRightX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 2 );
drawImage( upLeftY:botRightY, botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 3 );

drawImage( upLeftY+1:botRightY-1, botRightX-1, 1 ) = me.p(x)*black(1)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 1 );
drawImage( upLeftY+1:botRightY-1, botRightX-1, 2 ) = me.p(x)*black(2)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 2 );
drawImage( upLeftY+1:botRightY-1, botRightX-1, 3 ) = me.p(x)*black(3)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 3 );
```

```
  % West
  drawImage( upLeftY:botRightY, upLeftX, 1 ) = me.p(x)*white( 1 ) ...
    + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 1 );
  drawImage( upLeftY:botRightY, upLeftX, 2 ) = me.p(x)*white( 2 ) ...
    + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 2 );
  drawImage( upLeftY:botRightY, upLeftX, 3 ) = me.p(x)*white( 3 ) ...
    + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 3 );

  drawImage( upLeftY+1:botRightY-1, upLeftX+1, 1 ) = me.p(x)*black(1) ...
    + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 1 );
  drawImage( upLeftY+1:botRightY-1, upLeftX+1, 2 ) = me.p(x)*black(2) ...
    + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 2 );
  drawImage( upLeftY+1:botRightY-1, upLeftX+1, 3 ) = me.p(x)*black(3) ...
    + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 3 );

  % Check if missed event should be flipped so alpha will start being
  % reduced.
  if me.p(x) >= 1
    me.f(x) = 1;
  end

  % Increase or reduce the percentage of alpha by user-defined amount
  if me.f(x) == 0
    me.p(x) = me.p(x) + alpha;
  else
    me.p(x) = me.p(x) - alpha;
  end

  % Check to see if missed event should be erased from data structure
  if me.p(x) < 0.1 && me.f(x) == 1
    me.r(x) = [];
    me.c(x) = [];
    me.h(x) = [];
    me.w(x) = [];
    me.p(x) = [];
    me.f(x) = [];
  else

    % Increment x if not deleted
    x = x + 1;

  end

end

function [] = stillInterest(avifilename, writefilename, outputdir, ...
  motionCoords, datetimestring, copyDataDir)
%STILLINTEREST Detects still novel and missed events in the AVI stream.
%
%  Syntax
%
%    stillInterest(avifilename, writefilename, outputdir, ...
%    motionCoords, datetimestring, copyDataDir);
%
%  Description
%
```

```
%    STILLINTEREST creates a series of images with rectangles about
%    events of various colors, which depict the interest level of that
%    event.  Furthermore, STILLINTEREST creates black and white rectangles
%    about missed expected events.
%
%  Input
%
%    avifilename - directory and filename of where the input AVI file
%    exists.  Can be a relative or absolute path.
%
%    writefilename - output naming convention of all data files in the
%    subdirectories created in the output directories.
%
%    outputdir - output directory where further subdirectories will be
%    placed for processed output image files, the data structures for each
%    region, the interesting coordinates, the missed events data,
%    and a psuedo-color representation of the motion coordinate input.
%
%    motionCoords - a string of the directory and naming convention for
%    each motion coordinate for each frame.  For instance, the string
%    '..\MotionCoords\PeopleVanMotionCoords\PeopleVanMotionCoords' would
%    represent the motion coordinates reside in the
%    '..\MotionCoords\PeopleVanMotionCoords\' directory and each motion
%    coordinate file for each frame begins with 'PeopleVanMotionCoords'.
%    This means the files will be named as follows:
%
%      PeopleVanMotionCoords_1
%      PeopleVanMotionCoords_2
%      PeopleVanMotionCoords_3
%               ...
%      PeopleVanMotionCoords_X
%
%    Where X represents the last frame of the input video stream.  There
%    must be a one to one correlation of the motion coordinates to the
%    number of frames in the video.  During the algorithm, each structure
%    is loaded for each frame.  It is expected that each loaded motion
%    coordinate structure will load the variable named 'output'.  The
%    variable 'output' has the following structure:
%
%      output.L is connected components map
%      output.num is max components for the given map
%
%    Objects across multiple frames shall be tracked by using the same
%    unique number identifier across multiple motion coordinate maps.
%    Numbers may be reused in a given input video stream, but must be
%    unique for each frame and must not be reused unless there is at least
%    one frame as a buffer between the reuse of the number.
%
%    datetimestring - A string date and time of when this video begins.
%    The rate the time increases is determined by the
%    encoded frames per second in the AVI file.  Here
%    are some examples of date/time strings:
%      '26-Jul-2006 13:46:12'
%      '30-May-1982 06:17:30'
%      '01-Dec-2072 22:00:00'
%    They are in the form dd-mmm-yyyy hh:mm:ss.
%
```

```
%    copyDataDir - The directory to copy the memory data structures for
%    each pixel region from.  The .mat files will be copied to the 'data'
%    directory, underneath the provided output directory, as specified by
%    the 'outputdir' input variable.  This value may be an empty matrix if
%    you wish to create new memory data structures for the input video
%    stream.  Each regional data structure is an array of structures, which
%    adheres to the following specifications:
%
%      r - red intensity value
%      g - green intensity value
%      b - blue intensity value
%      h - height
%      w - width
%      vx - velocity x direction (not used in still, but it exists)
%      vy - velocity y direction (not used in still, but it exists)
%      t - time (hours from Saturday midnight)
%      m - membership level
%      d - the date of the last update of the group
%      f - the date of the last update and forget of the group
%      i - last interest value assigned by this group to an event
%
%   Each feature contains a mean and a variance value, except for 'm', 'd',
%   'f', and 'i'.  The variance represents the distance in only one
%   direction from the mean.  Each feature can be thought of as a triangle,
%   where the top point is defined by the mean and membership level.
%   Furthermore, the side points are defined by the variance away from the
%   mean and zero membership level.  However, for comparing events against
%   groups, the triangle is allowed to extend below the zero membership
%   level.
%
%  Output
%
%    No return arguments.  All output is written to the disk at the
%    location supplied by 'outputdir'.  In the 'outputdir' directory, four
%    subdirectories will be created.  The processed image files will be
%    in the 'video' subdirectory.  The group data structures, with
%    updated values, will appear in the 'data' subdirectory.  The locations
%    of the interesting events will appear in the 'interMap' subdirectory.
%    The pseudo-colored connected components map from the 'output' variable
%    will be in the 'label' subdirectory.  Any missed events that occurred
%    during the video run will appear in 'misEvent' subdirectory in a text
%    file.
%
%  Examples
%
%    stillInterest( '..\OriginalVideos\PeopleVan.avi', ...
%      'PeopleVan', ...
%      'Results\PeopleVan\MotionX2', ...
%      '..\MotionCoords\PeopleVanMotionCoords\PeopleVanMotionCoords', ...
%      '26-Jul-2006 12:55:00', ...
%      copyDir );
%
%    stillInterest( 'OriginalVideos\PeopleVanUltimate.avi', ...
%      'TestRunOne', ...
%      'Results\PeopleVanUltimate\Motion', ...
%      'PVUMotionCoords\PVUMC', ...
%      '9-Dec-2008 22:01:23', ...
```

```
%      [] );

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User Defined Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The seconds of video that appear before and after an interesting event.
frameBuff = 5;

% The floating point decimal between 0.0 and 1.0, inclusive.
% Events with interesting levels above or equal to this threshold
% will have a rectangle drawn about them and will be considered
% interesting.  All objects below this threshold will not be marked
% and will not be considered interesting.
interThresh = 0.75;

% Initial feature variances for new group
initRedVar = 0.03;
initGreenVar = 0.03;
initBlueVar = 0.03;
initHeightVar = 3;
initWidthVar = 4;
initTimeVar = 0.25;

% When a group is updated, this is the percentage that the mean of the
% existing group is altered to be more like the event that matches the
% group.
gUpdate = 0.1;

% When a group is updated, this is the percentage the variance is altered,
% whether that is outward or inward.
gVarMod = 0.01;

% When a group is updated, this is the upper bound of a feature, when
% compared to the group to consider to increase the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then increase the variance.
gUpBound = 0.95;

% When a group is updated, this is the lower bound of a feature, when
% compared to the group to consider to decrease the variance.  If the
% absolute difference of the feature from the group mean, all divided by
% the group variance is above this number, then decrease the variance.
gLoBound = 0.05;

% Each group becomes temporarily locked from updates after an update
% lockTime = 5.78703704e-006;  % half second in days
% lockTime = 1.15740741e-005;  % one second in days
lockTime = 3.47222222e-005;  % three seconds in days

% Membership increase rate for new group or group update
memIncRate = 0.1;
```

```
% How confident are we that this object belongs to the group? (-Inf to 1)
confidence = 0.75;

% Background learning rate per frame
bgAlpha = 0.014;

% Foreground learning rate per frame
fgAlpha = 0.1;

% The threshold after differencing the bg/fg models.
% If the difference is greater than this, then the bg is considered
% different from the fg.
diffThresh = 0.3;

% Type of forget function: 'step', 'linear', or 'quad'
ffType = 'step';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setup Before Main Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Show the user the percent of the video data has been processed every ten
% percent.
percentDone = 0.1;

% Discover the number of frames for the AVI file
movInfo = aviinfo( avifilename );

% Save the seconds per frame
secPerFrame = 1 / movInfo.FramesPerSecond;

% Convert to days per frame because we want this fraction for adding onto
% the serial date.
% days/frame = (sec/frame) * (1m/60s) * (1h/60m) * (1d/24h)
daysPerFrame = secPerFrame * 1.1620e-005;

% Save the number of frames to buffer interesting events
frameBuffer ...
  = round( frameBuff * movInfo.FramesPerSecond );

% Find all backslashes, if none found, just start at 1
begin = strfind( avifilename, '\' ) + 1;
if isempty( begin )
  begin = 1;
end

% Save the strings that point to the output directories
videoDir = [ outputdir, '\video\' ];
labelDir = [ outputdir, '\label\' ];
dataDir = [ outputdir, '\data\' ];
interDir = [ outputdir, '\interMap\' ];
misDir = [ outputdir, '\misEvent\' ];

% Create a directory to place the images into
%   May generate a warning if the directory already exists
mkdir( videoDir );
mkdir( labelDir );
```

```
mkdir( dataDir );
mkdir( interDir );
mkdir( misDir );

% Append specific name for data files
videoDir = [videoDir writefilename];
labelDir = [labelDir writefilename];
dataDir = [dataDir writefilename];
interDir = [interDir writefilename];
misDir = [misDir writefilename];

% Create a file for missed events
fidme = fopen( [misDir '_missedEvents.txt'], 'wt' );

% Get the number of 8x8 subimages for this movie sequence
%   Any extra pixels are shaved off
numVert  = floor( movInfo.Height / 8 );
numHoriz = floor( movInfo.Width / 8 );

% The total number of elements for one plane of intensity values
totalPixels = movInfo.Height * movInfo.Width;

% If empty matrix, create new data structures
if isempty(copyDataDir)

  data = struct( 'r', [], 'g', [], 'b', [], 'h', [], 'w', [], ...
    'vx', [], 'vy', [], 't', [], 'm', [], 'd', [], 'f', [], 'i', [] );

  % Create a .mat file for each 8x8 region
  for r = 1:numVert
    for c = 1:numHoriz
      save( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );
    end
  end

else

  % Copy from the given directory to data directory
  copyfile(copyDataDir, [outputdir '\data\']);

end

% The missed events data structure is for drawing black and white
% rectangles on the output video of where an event should have taken place
% and that event's approximate size.  The structure of missed events is as
% follows:
%   r - row of missed event in 'data'
%   c - column of missed event in 'data'
%   h - height of missed event
%   w - width of missed event
%   p - the percent alpha value of the rectangle drawn to screen
%   f - the flag, when it is zero 'p' is going up, when it is one 'p' is
%   going down.
% Missed event rectangles fade in and then fade out with a rate defined by
% the user in the 'drawmisevents' function.
misEvents = struct( 'r', [], 'c', [], 'h', [], 'w', [], 'p', [], 'f', [] );
```

```
% Convert the input date string to serial format
sTime = datenum( datetimestring );

% Display when the video starts when it was recorded
disp( ['Video begins at ' datestr(sTime, 0)] );

% Maintains the uninterrupted linear video sequence with the interesting
% event.
clip = 0;

% Maintains the frame number for each clip.
sequence = 1;

% Determines what frame to stop writing images to the disk before/after
% an interesting event.
postBuff = 0;

% Initialize the background and foreground model for detecting still events
movFrame = aviread( avifilename, 1 );
bgModel = im2double( movFrame.cdata );
fgModel = bgModel;

% Begin algorithm timer
tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start reading in from the input video
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Loop for every frame of the input movie
for i = 1:movInfo.NumFrames

  % Extract the current frame
  movFrame = aviread( avifilename, i );

  % Copy the cdata and convert to double
  newImage = im2double( movFrame.cdata );

  % Used to show interest levels
  drawImage = newImage;

  % Initialize the interesting event structure,
  % map - binary map, where zero represents no motion and one represents
  %       motion of the object
  % time - serial time/date of event
  % interest - the interesting value from 0 to 1, inclusive.  One is the
  %            highest level of interest.
  interMap = struct('map', [], 'time', [], 'interest', []);

  % Load this frame's motion coordinates into variable 'output'
  load([motionCoords '_' num2str(i)]);

  % For now, assume no interesting objects
  drewObject = 0;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Update non-motion regions for the background/foreground models
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Save where motion occurs to binary image
hitall = output.L ~= 0;

% Loop for each 8x8 pixel region
for g = 1:numVert
  for h = 1:numHoriz

    % Calculate the square we are looking at for this loop
    row = [ 1:8 ] + 8 * ( g - 1 );
    col = [ 1:8 ] + 8 * ( h - 1 );

    % Get region movement data
    hit = hitall(row,col);

    % Update models only if no movement in region
    % Update all or none of region
    if ~any( any( hit ) )

      fgModel(row,col,:) = ...
        ((1 - fgAlpha) * fgModel(row,col,:)) ...
        + (fgAlpha * newImage(row,col,:));

      bgModel(row,col,:) = ...
        ((1 - bgAlpha) * bgModel(row,col,:)) ...
        + (bgAlpha * newImage(row,col,:));

    end

  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Discover still events
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate difference between background and foreground models
sMap3 = abs( bgModel - fgModel ) >= diffThresh;

% OR the three planes together
sMap = sMap3(:,:,1) | sMap3(:,:,2) | sMap3(:,:,3);

% Perform an 8 directional connected components analysis
[sMap,num] = bwlabel(sMap,8);

% Extract serial date to vector date
vTime = datevec( sTime );

% Get the hours since Saturday midnight (beg. of Sunday)
hrs = (((weekday(datestr(sTime,1))-1)*24) + vTime(4) +...
  (vTime(5)/60) + (vTime(6)/3600));

% Loop for each still object found
for j = 1:num

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Record observables from still object
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Save the current object in motion
object = sMap == j;

% Get the indices of the current object
coords = find( object );

% Get the rows/cols of object
[row,col] = find( object );

% Calculate the center of the object (8x8 pixel regions)
r = ceil( floor( ( max( row ) + min( row ) ) / 2 ) / 8 );
c = ceil( floor( ( max( col ) + min( col ) ) / 2 ) / 8 );

% Calculate the mean of the intensities where the object was found
red = mean( newImage( coords ) );
green = mean( newImage( coords + totalPixels ) );
blue = mean( newImage( coords + 2 * totalPixels ) );

% Obtain the width/height of the still object
width = max( col ) - min( col ) + 1;
height = max( row ) - min( row ) + 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load the appropriate data region into 'data'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Store observable data into appropriate region
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Check to see if observed data matches previous data
match = 0;
besttest = 0;  % Only consider groups above this value
bestmember = 0;
ind = [];
bdta = 0;
bdtb = 0;
bdtc = 0;
x = 1;
b = [];
bestgroup = 0;

% Loop through all the groups
while x <= size(data.r,1)

  % Forget groups as necessary
  switch lower(ffType)
    case {'step'}
      [data,del] = fgStep( data, x, sTime );
    case {'linear'}
      [data,del] = fgLinear( data, x, sTime );
    case {'quad'}
```

```
    [data,del] = fgQuadratic( data, x, sTime );
  otherwise
    disp( 'Unkown forget type.  Defaulting to step.' );
    [data,del] = fgStep( data, x, sTime );
end

% If deleted/forgotten, continue searching
if del == 1
  continue;
end

% If the data is within proper variance
dr = abs( data.r(x,1) - red );
dg = abs( data.g(x,1) - green );
db = abs( data.b(x,1) - blue );
dh = abs( data.h(x,1) - height );
dw = abs( data.w(x,1) - width );
dta = abs( data.t(x,1) - hrs );

% Check for Sat to Sun rollover
% one week = 168 hours
dtb = data.t(x,1) - (hrs - 168);
dtc = (hrs + 168) - data.t(x,1);

% Get the minimum time as a check for both the Sat to Sun rollover
% and for normal time operations during the week
dtd = min( dtb, dtc );
dt = min( dta, dtd );

% Get membership for each observable [y=mx+b]
mr = -(data.m(x)/data.r(x,2))*dr + data.m(x);
mg = -(data.m(x)/data.g(x,2))*dg + data.m(x);
mb = -(data.m(x)/data.b(x,2))*db + data.m(x);
mh = -(data.m(x)/data.h(x,2))*dh + data.m(x);
mw = -(data.m(x)/data.w(x,2))*dw + data.m(x);
mt = -(data.m(x)/data.t(x,2))*dt + data.m(x);

% Combine observable membership.
% This is the membership level of the observable data in relation
% to the group.
member = mean( [mr mg mb mh mw mt], 2 );

% Divide by max group membership to standardize results
test = member / data.m(x);

% Is this the best match so far?
if test >= besttest

  % We found at least one match
  match = 1;

  % Maintain list of indicies
  ind(end+1) = x;

  % Test that confidence
  if test >= confidence
```

```
      % This event passes the confidence level for this group.  Save
      % the best group information to be updated with this event later.

      % Save best info for later
      bestmember = member;
      b = x;
      besttest = test;
      bdta = dta;
      bdtb = dtb;
      bdtc = dtc;

      % Don't bother looking for groups to get a relative interest
      % level, but not to update.  We have one that passed the
      % confidence level, so we will at least update this group.
      bestgroup = 1;

    elseif bestgroup == 0

      % We haven't found a group that has passed the confidence level,
      % but save this group because it is the best we have seen thus
      % far.  We will not update this group, but simply use it to
      % compare to the current event, which will let us define an
      % interest level for the event and a membership level for the new
      % group.

      % Save best info for later
      bestmember = member;
      b = x;
      besttest = test;
      bdta = dta;
      bdtb = dtb;
      bdtc = dtc;

    end

  end

  % Increment x
  x = x + 1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine if a match was found
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if match == 1 && bestgroup == 1

  % We found a similar group to update...

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Modify existing group
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   % Alter the center data point according to weight
  data.r(b,1) = (1-gUpdate)*data.r(b,1) + gUpdate*red;
  data.g(b,1) = (1-gUpdate)*data.g(b,1) + gUpdate*green;
```

```
data.b(b,1) = (1-gUpdate)*data.b(b,1) + gUpdate*blue;
data.h(b,1) = (1-gUpdate)*data.h(b,1) + gUpdate*height;
data.w(b,1) = (1-gUpdate)*data.w(b,1) + gUpdate*width;

% Select the smallest time difference
if bdtb <= bdta && bdtb <= bdtc
  % The hours must be moved to next week
  thrs = hrs - 168;
elseif bdtc <= bdta && bdtc <= bdtb
  % The hours must be moved to prev week
  thrs = hrs + 168;
else
  % Same week
  thrs = hrs;
end

data.t(b,1) = (1-gUpdate)*data.t(b,1) + gUpdate*thrs;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modify variance if new observation in extreme middle or
% extreme outside of group.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (abs( red - data.r(b,1) ) / data.r(b,2)) > gUpBound
  % Increase variance
  data.r(b,2) = (1+gVarMod)*data.r(b,2);
elseif (abs( red - data.r(b,1) ) / data.r(b,2)) ...
    < gLoBound
  % Decrease variance
  data.r(b,2) = (1-gVarMod)*data.r(b,2);
end

if (abs( green - data.g(b,1) ) / data.g(b,2)) > gUpBound
  % Increase variance
  data.g(b,2) = (1+gVarMod)*data.g(b,2);
elseif (abs( green - data.g(b,1) ) / data.g(b,2)) ...
    < gLoBound
  % Decrease variance
  data.g(b,2) = (1-gVarMod)*data.g(b,2);
end

if (abs( blue - data.b(b,1) ) / data.b(b,2)) > gUpBound
  % Increase variance
  data.b(b,2) = (1+gVarMod)*data.b(b,2);
elseif (abs( blue - data.b(b,1) ) / data.b(b,2)) ...
    < gLoBound
  % Decrease variance
  data.b(b,2) = (1-gVarMod)*data.b(b,2);
end

if (abs( height - data.h(b,1) ) / data.h(b,2)) > gUpBound
  % Increase variance
  data.h(b,2) = (1+gVarMod)*data.h(b,2);
elseif (abs( height - data.h(b,1) ) / data.h(b,2)) ...
    < gLoBound
  % Decrease variance
  data.h(b,2) = (1-gVarMod)*data.h(b,2);
```

```
end

if (abs( width - data.w(b,1) ) / data.w(b,2)) > gUpBound
  % Increase variance
  data.w(b,2) = (1+gVarMod)*data.w(b,2);
elseif (abs( width - data.w(b,1) ) / data.w(b,2)) ...
    < gLoBound
  % Decrease variance
  data.w(b,2) = (1-gVarMod)*data.w(b,2);
end

if (abs( thrs - data.t(b,1) ) / data.t(b,2)) > gUpBound
  % Increase variance
  data.t(b,2) = (1+gVarMod)*data.t(b,2);
elseif (abs( thrs - data.t(b,1) ) / data.t(b,2)) ...
    < gLoBound
  % Decrease variance
  data.t(b,2) = (1-gVarMod)*data.t(b,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ensure proper hours per week
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Fix the mean time if a rollover occurs across Sat and Sun
if data.t(b,1) < 0
  % Move to prev week
  data.t(b,1) = data.t(b,1) + 168;
elseif data.t(b,1) > 168
  % Move to next week
  data.t(b,1) = data.t(b,1) - 168;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine if group is locked
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Is this group locked temporarily?
if data.d(b) + lockTime < sTime

  % If not locked...

  % Increase the membership (height)
  data.m(b) = data.m(b) + memIncRate;

  % Remember last assigned membership
  data.i(b) = bestmember;

  % Set last update date/time (for forgetting and lockout)
  data.d(b) = sTime;
  data.f(b) = sTime;

else

  % If locked...

  % Use last assigned color for this event
```

```
    bestmember = data.i(b);

  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Merge similar groups
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  data = checkgroups(data,ind,'still');

elseif match == 1 && bestgroup == 0

  % We found the most similar group, but it does not pass the
  % confidence level.  We will create a new group, but the membership
  % will be based off the related interest level of the event to this
  % group we found.

  % Place group at end of array
  z = size(data.r,1)+1;

  % Set the group mean value
  data.r(z,1) = red;
  data.g(z,1) = green;
  data.b(z,1) = blue;
  data.h(z,1) = height;
  data.w(z,1) = width;
  data.vx(z,1) = 0;  % Not used for still interest
  data.vy(z,1) = 0;  % Not used for still interest
  data.t(z,1) = hrs;

  % Set the initial variance
  data.r(z,2) = initRedVar;
  data.g(z,2) = initGreenVar;
  data.b(z,2) = initBlueVar;
  data.h(z,2) = initHeightVar;
  data.w(z,2) = initWidthVar;
  data.vx(z,2) = 0;  % Not used for still interest
  data.vy(z,2) = 0;  % Not used for still interest
  data.t(z,2) = initTimeVar;

  % Set initial membership to max of init membership or bestmember
  bestmember = max( bestmember, memIncRate );
  data.m(z) = bestmember;

  % Set last update date/time (for forgetting and lockout)
  data.d(z) = sTime;
  data.f(z) = sTime;

  % Remember last assigned membership.
  % But subtract the membership increase rate because this is the
  % reported interest level.
  bestmember = bestmember - memIncRate;
  data.i(z) = bestmember;

else

  % No groups similar enough to this event...
```

```
  %%%%%%%%%%%%%%%%%%%%%%%%%%
  % Create a new group
  %%%%%%%%%%%%%%%%%%%%%%%%%%

  % Place group at end of array
  b = size(data.r,1)+1;

  % Set the group mean value
  data.r(b,1) = red;
  data.g(b,1) = green;
  data.b(b,1) = blue;
  data.h(b,1) = height;
  data.w(b,1) = width;
  data.vx(b,1) = 0;  % Not used for still interest
  data.vy(b,1) = 0;  % Not used for still interest
  data.t(b,1) = hrs;

  % Set the initial variance
  data.r(b,2) = initRedVar;
  data.g(b,2) = initGreenVar;
  data.b(b,2) = initBlueVar;
  data.h(b,2) = initHeightVar;
  data.w(b,2) = initWidthVar;
  data.vx(b,2) = 0;  % Not used for still interest
  data.vy(b,2) = 0;  % Not used for still interest
  data.t(b,2) = initTimeVar;

  % Set initial membership
  data.m(b) = memIncRate;
  bestmember = 0;

  % Set last update date/time (for forgetting and lockout)
  data.d(b) = sTime;
  data.f(b) = sTime;

  % Remember last assigned membership
  data.i(b) = bestmember;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Draw the interest color of the event
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Draw the rectangle if above interesting threshold
if (1-bestmember) >= interThresh

  % Draw a box around the object according to membership
  drawImage = drawObject( drawImage, row, col, bestmember );

  % Set the flag that we drew an object
  drewObject = 1;

  % Save to the interesting event map
  index = numel( interMap.time ) + 1;
  interMap(index).map = object;
```

```
    interMap(index).time = sTime;
    interMap(index).interest = (1-bestmember);

  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Save the data structure to disk
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  save( [dataDir '_data_' num2str(r) '_' num2str(c)], 'data' );

end  % Loop to the next event in the scene...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Record and draw missed expected events
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Write missed events to text file and return 'pointers' of missed
% events in the data structure
misEvents = missedgroups(dataDir,numVert,numHoriz,daysPerFrame,...
  sTime,hrs,fidme,misEvents);

% Draw missed events
if ~isempty(misEvents.r)
  [drawImage,misEvents] = drawmisevents(drawImage,movInfo,misEvents);

  % Write this image to disk
  drewObject = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Write images to disk for the current frame and buffering frames
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% If we drew a rectangle indicating an interesting event
if i <= postBuff || drewObject == 1

  % If the current frame is beyond the post buffer
  if i > postBuff

    % Write previous input movie frames
    preBuff = i - frameBuffer;

    % Check for beginning of movie
    if preBuff < 1
      preBuff = 1;
    end

    % Continue the last clip because the buffers overlap
    if preBuff < postBuff
      preBuff = postBuff + 1;
    else
      % Go to next clip, reset sequence
      clip = clip + 1;
      sequence = 1;
    end
```

```
      for x = preBuff:(i-1)
        % Extract the frame
        tmpFrame = aviread( avifilename, x );

        % Copy the cdata and convert to double
        tmpImage = im2double( tmpFrame.cdata );

        % Write the file
        imwrite( tmpImage, [ videoDir, ...
          '.', num2str( clip ), '.', num2str( sequence ), '.jpg'] );

        sequence = sequence + 1;
      end

    end

    % Write this interesting event to the disk
    imwrite( drawImage, [ videoDir, ...
      '.', num2str( clip ), '.', num2str( sequence ), '.jpg'] );

    sequence = sequence + 1;

    % If interesting objects in this frame
    if drewObject == 1

      % Save the frame where the Post Buffer would end
      postBuff = i + frameBuffer;

    end

  end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % End of the main loop cleanup
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Save the interesting events to disk for this frame
  save( [interDir '_interMap_' num2str(i)], 'interMap' );

  % Write the psuedo-color motion coords to disk
  imwrite( label2rgb( output.L ), [ labelDir ...
    , '.', num2str( i ), '.jpg' ] );

  % Increment the time with days per frame
  sTime = sTime + daysPerFrame;

  % Progress report every ten percent
  if abs( ( i / movInfo.NumFrames ) - percentDone ) < 0.01
    disp( [ num2str( percentDone * 100 ), '% complete.' ] );
    percentDone = percentDone + 0.1;
  end

end % Loop to the next input video frame...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Cleanup after main loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Last check to forget groups
forgetgroups(dataDir,numVert,numHoriz,sTime,ffType);

% Say when the video time ends when the video was recorded
disp( ['Video ends at ' datestr(sTime, 0)] );

% Close writing to the missed events text file
status = fclose(fidme);

% Report if error
if status == -1
  disp( 'Unable to close the missed events file.' );
end

% Print out algorithm time
n = toc;
disp( [ num2str( n / 60 ), ' minutes to complete algorithm.' ] );



function [ drawImage ] = drawObject( drawImage, row, col, hsvColor )
% DRAWOBJECT Draws a rectangle at the given coordinates.

% Save the beginning/ending rows/columns for the box
upLeftX = min( col );
upLeftY = min( row );
botRightX = max( col );
botRightY = max( row );

% Convert HSV color to RGB
rgbColor = hsv2rgb( cat( 3, hsvColor, 1.0, 1.0 ) );

%%%%%%%%%%%%%%%%%%%%%%%%
% Draw the bounding box
%%%%%%%%%%%%%%%%%%%%%%%%

% North
drawImage( upLeftY, upLeftX:botRightX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( upLeftY, upLeftX:botRightX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( upLeftY, upLeftX:botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% South
drawImage( botRightY, upLeftX:botRightX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( botRightY, upLeftX:botRightX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( botRightY, upLeftX:botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% East
drawImage( upLeftY:botRightY, botRightX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( upLeftY:botRightY, botRightX, 2 ) = ...
```

```
  rgbColor( 1, 1, 2 );
drawImage( upLeftY:botRightY, botRightX, 3 ) = ...
  rgbColor( 1, 1, 3 );

% West
drawImage( upLeftY:botRightY, upLeftX, 1 ) = ...
  rgbColor( 1, 1, 1 );
drawImage( upLeftY:botRightY, upLeftX, 2 ) = ...
  rgbColor( 1, 1, 2 );
drawImage( upLeftY:botRightY, upLeftX, 3 ) = ...
  rgbColor( 1, 1, 3 );



function [drawImage,me] = drawmisevents(drawImage,movInfo,me)
% DRAWMISEVENTS Draws black and white rectangles around missed events.

% User-defined fade in/out of missed events
alpha = 0.1;

% Loop for every missed event
x = 1;
while x <= numel(me.r)

  % Save the beginning/ending rows/columns for the box
  upLeftX = (me.c(x)*8-4) - ceil(me.w(x)/2);
  upLeftY = (me.r(x)*8-4) - ceil(me.h(x)/2);
  botRightX = (me.c(x)*8-4) + ceil(me.w(x)/2);
  botRightY = (me.r(x)*8-4) + ceil(me.h(x)/2);

  % Limit the boundaries to the output video
  if upLeftX < 1
    upLeftX = 1;
  end
  if upLeftY < 1
    upLeftY = 1;
  end
  if botRightX > movInfo.Width
    botRightX = movInfo.Width;
  end
  if botRightY > movInfo.Height
    botRightY = movInfo.Height;
  end

  % RGB colors
  white = [1 1 1];
  black = [0 0 0];

  %%%%%%%%%%%%%%%%%%%%%%%%
  % Draw the bounding box
  %%%%%%%%%%%%%%%%%%%%%%%%

  % North
  drawImage( upLeftY, upLeftX:botRightX, 1 ) = me.p(x)*white( 1 ) ...
    + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 1 );
  drawImage( upLeftY, upLeftX:botRightX, 2 ) = me.p(x)*white( 2 ) ...
    + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 2 );
```

```
drawImage( upLeftY, upLeftX:botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( upLeftY, upLeftX:botRightX, 3 );

drawImage( upLeftY+1, upLeftX+1:botRightX-1, 1 ) = me.p(x)*black(1) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 1 );
drawImage( upLeftY+1, upLeftX+1:botRightX-1, 2 ) = me.p(x)*black(2) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 2 );
drawImage( upLeftY+1, upLeftX+1:botRightX-1, 3 ) = me.p(x)*black(3) ...
  + (1-me.p(x))*drawImage( upLeftY+1, upLeftX+1:botRightX-1, 3 );

% South
drawImage( botRightY, upLeftX:botRightX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 1 );
drawImage( botRightY, upLeftX:botRightX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 2 );
drawImage( botRightY, upLeftX:botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( botRightY, upLeftX:botRightX, 3 );

drawImage( botRightY-1, upLeftX+1:botRightX-1, 1 ) = me.p(x)*black(1)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 1 );
drawImage( botRightY-1, upLeftX+1:botRightX-1, 2 ) = me.p(x)*black(2)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 2 );
drawImage( botRightY-1, upLeftX+1:botRightX-1, 3 ) = me.p(x)*black(3)...
  + (1-me.p(x))*drawImage( botRightY-1, upLeftX+1:botRightX-1, 3 );

% East
drawImage( upLeftY:botRightY, botRightX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 1 );
drawImage( upLeftY:botRightY, botRightX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 2 );
drawImage( upLeftY:botRightY, botRightX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, botRightX, 3 );

drawImage( upLeftY+1:botRightY-1, botRightX-1, 1 ) = me.p(x)*black(1)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 1 );
drawImage( upLeftY+1:botRightY-1, botRightX-1, 2 ) = me.p(x)*black(2)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 2 );
drawImage( upLeftY+1:botRightY-1, botRightX-1, 3 ) = me.p(x)*black(3)...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, botRightX-1, 3 );

% West
drawImage( upLeftY:botRightY, upLeftX, 1 ) = me.p(x)*white( 1 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 1 );
drawImage( upLeftY:botRightY, upLeftX, 2 ) = me.p(x)*white( 2 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 2 );
drawImage( upLeftY:botRightY, upLeftX, 3 ) = me.p(x)*white( 3 ) ...
  + (1-me.p(x))*drawImage( upLeftY:botRightY, upLeftX, 3 );

drawImage( upLeftY+1:botRightY-1, upLeftX+1, 1 ) = me.p(x)*black(1) ...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 1 );
drawImage( upLeftY+1:botRightY-1, upLeftX+1, 2 ) = me.p(x)*black(2) ...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 2 );
drawImage( upLeftY+1:botRightY-1, upLeftX+1, 3 ) = me.p(x)*black(3) ...
  + (1-me.p(x))*drawImage( upLeftY+1:botRightY-1, upLeftX+1, 3 );

% Check if missed event should be flipped so alpha will start being
% reduced.
```

```
  if me.p(x) >= 1
    me.f(x) = 1;
  end

  % Increase or reduce the percentage of alpha by user-defined amount
  if me.f(x) == 0
    me.p(x) = me.p(x) + alpha;
  else
    me.p(x) = me.p(x) - alpha;
  end

  % Check to see if missed event should be erased from data structure
  if me.p(x) < 0.1 && me.f(x) == 1
    me.r(x) = [];
    me.c(x) = [];
    me.h(x) = [];
    me.w(x) = [];
    me.p(x) = [];
    me.f(x) = [];
  else

    % Increment x if not deleted
    x = x + 1;

  end

end

function [me] = missedgroups(datafiles,mr,mc,timestep,sTime,hrs,fid,me)
% MISSEDGROUPS Determine if an expected event did not occur.
%
%  Syntax
%
%    [me] = missedgroups(datafiles,mr,mc,timestep,sTime,hrs,fid,me);
%
%  Description
%
%    MISSEDGROUPS loops through each regional memory data structure and
%    determines if a group time variance window has just expired
%    within the last time step, if that same group has a membership
%    level of above a user-defined point, and if that group was not updated
%    within the expected time frame.  If a group passes these three
%    stipulations, then a missed event has occured.  MISSEDGROUPS returns
%    the missed event data structure, which is also an input argument.
%
%  Input
%
%    datafiles - The directory and naming convention string of the regional
%    memory data structures, which are defined and documented in
%    'motionInterest.m'.  The following are possible strings:
%
%      'Results\PeopleVan\Motion\data\PeopleVan'
%      'Results\Tennis\Still\data\Tennis'
%      'Results\Walking\Motion\data\Walking'
%
%    The regional memory data structures should have the following naming
%    convention:
```

```
%
%      'FILENAME_data_R_C.mat'
%
%    Where FILENAME is the name of the input AVI video stream, such as
%    'PeopleVan', 'Tennis', or 'Walking'.  R represents the row of the
%    region and C represents the column of the region, such as 4, 5, 8, 20,
%    30, or 45.
%
%    mr - The number of rows of regional memory data structures.
%
%    mc - The number of columns of regional memory data structures.
%
%    timestep - This is the amount of time that has passed since the last
%    call to MISSEDGROUPS.  This variable must be in terms of days.
%    So if 'timestep' is equal to 1.75, it would represent one and
%    three-quarters of a day have passed since the last time this function
%    has been was called.  This value is not CPU time, but time that has
%    passed in the video.
%
%    sTime - The current serial date and time in the input video stream.
%
%    hrs - The current hours in the week since Midnight of Saturday.
%
%    fid - The Matlab file identifier obtained with the 'fopen' command.
%    MISSEDGROUPS records the missed event information to this text file.
%
%    me - The missed event data structure as defined and documented in
%    'motionNovelty.m'.
%
%  Output
%
%    me - The same missed event data structure, but with the possibility of
%    newly discovered missed expected events added.
%
%  Examples
%
%    daysPerFrame = secPerFrame * 1.1620e-005;
%    hrs = 1;
%    fid = fopen( 'missedEvents.txt' ], 'wt' );
%    sTime = datenum('8-Oct-2006 01:00:00');
%    dataDir = 'Results\PeopleVan\Motion\data\PeopleVan';
%    mr = 30;
%    mc = 45;
%    [me] = missedgroups(dataDir,mr,mc,daysPerFrame,sTime,hrs,fid,me);

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

% User-defined minimum membership level for expected events.
mcheck = 0.8;

% Loop for each row/col in data
for r = 1:mr
  for c = 1:mc

    % Load the region into memory
```

```
    load( [datafiles '_data_' num2str(r) '_' num2str(c)], 'data' );

    % Check if any exist at this region
    if isempty(data.r)
      continue;
    end

    % Discover events that just passed having any membership level
    tia = hrs - (data.t(:,1) + data.t(:,2));

    % Check Sat to Sun rollover
    tib = (hrs + 168) - (data.t(:,1) + data.t(:,2));

    % Choose the minimum of the two
    ti = min( tia, tib );

    % Discover events that occured outside the expected times
    % If the date is less then the early side of the time group
    di = data.d(:) < (sTime - (data.t(:,2)/12) - timestep);

    % Discover events that have appropriate membership levels
    mi = data.m(:) >= mcheck;

    % Logical AND these together to discover missed events
    tf = find( ( (tia > 0 & tia <= (timestep*24)) | ...
      (tib > 0 & tib <= (timestep*24)) ) & di & mi );

    % If we found missed events
    if ~isempty(tf)

      % Append the newly discovered missed events
      b = numel(me.r) + 1;
      e = b + numel(tf) - 1;
      me.r(b:e) = r;
      me.c(b:e) = c;
      me.h(b:e) = data.h(tf);
      me.w(b:e) = data.w(tf);
      me.p(b:e) = 0.1;
      me.f(b:e) = 0;

      % Write missed events to disk
      for x = tf'
        fprintf( fid, '%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f\n', ...
          data.r(x,1), data.r(x,2), ...
          data.g(x,1), data.g(x,2), ...
          data.b(x,1), data.b(x,2), ...
          data.h(x,1), data.h(x,2), ...
          data.w(x,1), data.w(x,2), ...
          data.t(x,1), data.t(x,2), ...
          data.m(x), ...
          data.d(x) );
      end

    end

  end
end
```

```
function [data] = mergegroups(data,h,g,roll)
% MERGEGROUPS Merge group h into group g.
%
%  Syntax
%
%    data = mergegroups(data,h,g,roll);
%
%  Description
%
%    MERGEGROUPS merges group h into group g.  When this function returns,
%    group h has been removed and group g has been replaced with the
%    updated group.  The updated group covers the same feature variance
%    space as previously covered by both the h and g groups.  The new group
%    membership level is the maximum of h and g.  The feature means of the
%    new group is a ratio of groups h and g, based upon the membership
%    level of both of those groups.  The higher the membership level of a
%    group, the more influence it has on the new group feature mean.  Group
%    h and group g must never be the same.
%
%  Input
%
%    data - One region of the regional memory data structures, which
%    is defined and documented in 'motionInterest.m'.
%
%    h - The index into the 'data' variable of where group h exists.
%
%    g - The index into the 'data' variable of where group g exists.
%
%    roll - Roll specifies the relation of timing with respect to the
%    group g.  If roll is zero, do not worry about the Sat/Sun hour
%    crossover.  If roll is one, add 168 hours onto group h to match
%    group g.  If roll is two, subtract 168 hours from group h to match
%    group g.  This is needed because a sudden change in the hours in the
%    week occurs between Saturday night and Sunday morning.
%
%  Output
%
%    data - The same region of the regional memory data structures.
%    However, this output will have one less group and one updated group.
%
%  Examples
%
%    g = 4;
%    h = 8;
%    roll = 0;
%    data = mergegroups(data,g,h,roll);

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

% Group membership level affects the influence the group has on the new
% group mean.  Calculate the total membership between the two groups.
tm = data.m(g) + data.m(h);

% Calculate the new mean based on each groups influence from group
```

```
% membership levels.
red = ( data.r(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.r(h,1) * ( data.m(h) / tm ) );
green = ( data.g(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.g(h,1) * ( data.m(h) / tm ) );
blue = ( data.b(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.b(h,1) * ( data.m(h) / tm ) );
height = ( data.h(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.h(h,1) * ( data.m(h) / tm ) );
width = ( data.w(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.w(h,1) * ( data.m(h) / tm ) );
velx = ( data.vx(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.vx(h,1) * ( data.m(h) / tm ) );
vely = ( data.vy(g,1) * ( data.m(g) / tm ) ) ...
  + ( data.vy(h,1) * ( data.m(h) / tm ) );

if roll == 0

  % Normal timing
  time = ( data.t(g,1) * ( data.m(g) / tm ) ) ...
    + ( data.t(h,1) * ( data.m(h) / tm ) );

elseif roll == 1

  % Move h back to previous week, where g is
  time = ( data.t(g,1) * ( data.m(g) / tm ) ) ...
    + ( (data.t(h,1)+168) * ( data.m(h) / tm ) );

elseif roll == 2

  % Move h ahead to next week, where g is
  time = ( data.t(g,1) * ( data.m(g) / tm ) ) ...
    + ( (data.t(h,1)-168) * ( data.m(h) / tm ) );

else

  % roll is invalid
  error( [ 'A roll value of ' num2str(roll) ' is invalid.' ] );

end

% The new group variance must cover the same area covered previously by
% both groups in this merger.  First, calculate the difference between the
% new group mean and the old group mean, for both groups.  Furthermore, add
% onto this difference the old group variance.  This calculation gives us a
% measure of what the variance should be of the new group to cover the same
% area as covered by both groups and slightly more.
drg = abs( red - data.r(g,1) ) + data.r(g,2);
dgg = abs( green - data.g(g,1) ) + data.g(g,2);
dbg = abs( blue - data.b(g,1) ) + data.b(g,2);
dhg = abs( height - data.h(g,1) ) + data.h(g,2);
dwg = abs( width - data.w(g,1) ) + data.w(g,2);
dvxg = abs( velx - data.vx(g,1) ) + data.vx(g,2);
dvyg = abs( vely - data.vy(g,1) ) + data.vy(g,2);
dtg = abs( time - data.t(g,1) ) + data.t(g,2);

drh = abs( red - data.r(h,1) ) + data.r(h,2);
```

```
dgh = abs( green - data.g(h,1) ) + data.g(h,2);
dbh = abs( blue - data.b(h,1) ) + data.b(h,2);
dhh = abs( height - data.h(h,1) ) + data.h(h,2);
dwh = abs( width - data.w(h,1) ) + data.w(h,2);
dvxh = abs( velx - data.vx(h,1) ) + data.vx(h,2);
dvyh = abs( vely - data.vy(h,1) ) + data.vy(h,2);

if roll == 0

  % Normal timing
  dth = abs( time - data.t(h,1) ) + data.t(h,2);

elseif roll == 1

  % Move h back to previous week, where g is
  dth = abs( time - (data.t(h,1)+168) ) + data.t(h,2);

elseif roll == 2

  % Move h ahead to next week, where g is
  dth = abs( time - (data.t(h,1)-168) ) + data.t(h,2);

end

% Second, determine the maximum distance for each feature and keep the
% largest, which will be the new group variance for that feature.
dr = max( drg, drh );
dg = max( dbg, dbh );
db = max( dgg, dgh );
dh = max( dhg, dhh );
dw = max( dwg, dwh );
dvx = max( dvxg, dvxh );
dvy = max( dvyg, dvyh );
dt = max( dtg, dth );

% Third, we want the new group to have the largest mean of the two merging
% groups.
newMem = max( data.m(h), data.m(g) );

% Fourth, get the latest dates and assigned membership levels.  This will
% be transfered to the new group.
dd = max( data.d(g), data.d(h) );
df = max( data.f(g), data.f(h) );
di = max( data.i(g), data.i(h) );

% Finally, we have the information to create a new group.  Replace group g
% with the new group and remove h.
data.r(g,1) = red;
data.g(g,1) = green;
data.b(g,1) = blue;
data.h(g,1) = height;
data.w(g,1) = width;
data.vx(g,1) = velx;
data.vy(g,1) = vely;
data.t(g,1) = time;
data.m(g) = newMem;
data.d(g) = dd;
```

```
data.f(g) = df;
data.i(g) = di;

% Set the new variance
data.r(g,2) = dr;
data.g(g,2) = dg;
data.b(g,2) = db;
data.h(g,2) = dh;
data.w(g,2) = dw;
data.vx(g,2) = dvx;
data.vy(g,2) = dvy;
data.t(g,2) = dt;

% Check for hours rollover
if data.t(g,1) < 0

  % Move to previous week
  data.t(g,1) = data.t(g,1) + 168;

elseif data.t(g,1) >= 168

  % Move to next week
  data.t(g,1) = data.t(g,1) - 168;

end

% Remove group h
data.r(h,:) = [];
data.g(h,:) = [];
data.b(h,:) = [];
data.h(h,:) = [];
data.w(h,:) = [];
data.vx(h,:) = [];
data.vy(h,:) = [];
data.t(h,:) = [];
data.m(h) = [];
data.d(h) = [];
data.f(h) = [];
data.i(h) = [];

function [] = forgetgroups(datafiles,mr,mc,sTime,type)
% FORGETGROUPS Groups not updated recently lose membership.
%
%  Syntax
%
%    forgetgroups(datafiles,mr,mc,sTime,type);
%
%  Description
%
%    FORGETGROUPS scans through all the groups in each region and calls the
%    appropriate forget function, which is determined by the variable
%    'type'.
%
%  Input
%
%    datafiles - The directory and naming convention string of the regional
%    memory data structures, which are defined and documented in
```

```
%    'motionInterest.m'.  The following are possible strings:
%
%      'Results\PeopleVan\Motion\data\PeopleVan'
%      'Results\Tennis\Still\data\Tennis'
%      'Results\Walking\Motion\data\Walking'
%
%    The regional memory data structures should have the following naming
%    convention:
%
%      'FILENAME_data_R_C.mat'
%
%    Where FILENAME is the name of the input AVI video stream, such as
%    'PeopleVan', 'Tennis', or 'Walking'.  R represents the row of the
%    region and C represents the column of the region, such as 4, 5, 8, 20,
%    30, or 45.
%
%    mr - The number of rows of regional memory data structures.
%
%    mc - The number of columns of regional memory data structures.
%
%    sTime - The current serial date and time of the video.  This variable
%    is compared against the dates stored within the groups in the regional
%    memory data structures.
%
%    type - May be any of the following strings:
%
%      'step' - A step-wise forget function
%      'linear' - A linear forget function
%      'quad' - A quadratic forget function
%
%    The type defaults to the step-wise forget function.
%
%  Output
%
%    Saves the regional memory data structures back to the disk with the
%    partially or fully forgotten groups.
%
%  Examples
%
%    mr = 30;
%    mc = 45;
%    sTime = datenum('19-Feb-2008 00:00:10');
%    datafiles = dataDir = 'Results\PeopleVan\Motion\data\PeopleVan';
%    type = 'quad';
%    forgetgroups(datafiles,mr,mc,sTime,type);

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

% Loop through the data structure
for r = 1:mr
  for c = 1:mc

    load( [datafiles '_data_' num2str(r) '_' num2str(c)], 'data' );

    x = 1;
```

```
    while x < size(data.r,1)

      % Determine type of forgetting to use
      switch lower(type)
        case {'step'}
          [data,del] = fgStep( data, x, sTime );
        case {'linear'}
          [data,del] = fgLinear( data, x, sTime );
        case {'quad'}
          [data,del] = fgQuadratic( data, x, sTime );
        otherwise
          disp( 'Unkown forget type.  Defaulting to step.' );
          [data,del] = fgStep( data, x, sTime );
      end

      % If we deleted the group, do not increment x
      if del ~= 1
        x = x + 1;
      end

    end

    save( [datafiles '_data_' num2str(r) '_' num2str(c)], 'data' );

  end
end

function [region,del] = fgStep(region,h,sTime)
% FGSTEP Forget this group stepwise if not updated recently.
%
%  Syntax
%
%    [region,del] = fgStep(region,h,sTime);
%
%  Description
%
%    FGSTEP subtracts from a group's membership for every
%    user-defined time interval that has passed since the group has been
%    last updated.  User-definitions include the time interval and the
%    amount of membership to subtract.
%
%  Input
%
%    region - A list of groups for a given region in the memory
%    data structure, which is defined and documented in 'motionInterest.m'.
%
%    h - The index into the variable 'region' to get the exact group.  The
%    group is determined to be partially forgotten, fully forgotten, or
%    left alone.
%
%    sTime - The current serial date and time of the video.  This variable
%    is compared against the date stored within the group.
%
%  Output
%
%    region - The same data structure given as input.  However, this
%    output may have one less groups due to forgetting.
```

```
%
%    del - Will be equal to zero if the group was not deleted.  Will be
%    equal to one if the group was deleted.
%
%  Examples
%
%    x = 3;
%    sTime = datenum('19-Feb-2008 00:00:10');
%    [data,del] = fgStep( data, x, sTime );

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

% Days after event is partially forgotten
% Can be partial days (1.75 or 0.2)
days = 14;

% Amount of membership to forget
fm = 0.5;

% Assume we will not delete this group
del = 0;

% Forget for each interval
d = (sTime - region.f(h)) / days;
x = floor(d);

% Save remainder for next forget time
r = d - x;

% Save the times forgotten
z = x;

% If time to partially forget this group
while x > 0

  x = x - 1;

  % Lower the membership of the group
  region.m(h) = region.m(h) - fm;

end

% If at least one forget interval
if z > 0

  % Remove group if no membership
  if region.m(h) <= 0

    region.r(h,:) = [];
    region.g(h,:) = [];
    region.b(h,:) = [];
    region.h(h,:) = [];
    region.w(h,:) = [];
    region.vx(h,:) = [];
    region.vy(h,:) = [];
```

```
    region.t(h,:) = [];
    region.m(h) = [];
    region.d(h) = [];
    region.f(h) = [];
    region.i(h) = [];

    % We deleted
    del = 1;

  else

    % Set the last update date/time flag (with remainder)
    region.f(h) = sTime - r;

  end

end

function [region,del] = fgLinear(region,h,sTime)
% FGLINEAR Forget this group linearly with time.
%
%  Syntax
%
%    [region,del] = fgLinear(region,h,sTime);
%
%  Description
%
%    FGLINEAR subtracts a linear amount from a group's membership as a
%    function of time that has passed since the group has been
%    last updated.  User-definitions include the time interval and the
%    membership decrement point to create the slope.
%
%  Input
%
%    region - A list of groups for a given region in the memory
%    data structure, which is defined and documented in 'motionInterest.m'.
%
%    h - The index into the variable 'region' to get the exact group.
%
%    sTime - The current serial date and time of the video.  This variable
%    is compared against the date stored within the group.
%
%  Output
%
%    region - The same data structure given as input.  However, this
%    output may have one less groups due to forgetting.
%
%    del - Will be equal to zero if the group was not deleted.  Will be
%    equal to one if the group was deleted.
%
%  Examples
%
%    x = 3;
%    sTime = datenum('19-Feb-2008 00:00:10');
%    [data,del] = fgLinear( data, x, sTime );

% Jeremy Paskali : Ver 0.0 : 09-10-2006
```

```
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

%%%%%%%%%%%%%%%%%%%%%%%%%
% User-definitions
%%%%%%%%%%%%%%%%%%%%%%%%%

% Days to forget a standard membership unit.
days = 7;

% Amount of membership to forget, which is the standard membership unit.
fm = 0.05;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform the membership subtraction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Slope is defined by the membership to forget over the days
slope = -fm / days;

% Linear forget function (y = mx + b):
%   y -> new membership level
%   m -> slope
%   x -> absolute delta time (sTime should be greater)
%   b -> old membership level
region.m(h) = (slope * abs(sTime - region.f(h))) + region.m(h);

% Assume we will not delete this group
del = 0;

% Remove group if no membership
if region.m(h) <= 0

  region.r(h,:) = [];
  region.g(h,:) = [];
  region.b(h,:) = [];
  region.h(h,:) = [];
  region.w(h,:) = [];
  region.vx(h,:) = [];
  region.vy(h,:) = [];
  region.t(h,:) = [];
  region.m(h) = [];
  region.d(h) = [];
  region.f(h) = [];
  region.i(h) = [];

  % We deleted
  del = 1;

else

  % Set the last update date/time flag
  region.f(h) = sTime;

end

function [region,del] = fgQuadratic(region,h,sTime)
```

```
% FGQUADRATIC Forget this group quadratically with time.
%
%  Syntax
%
%    [region,del] = fgQuadratic(region,h,sTime);
%
%  Description
%
%    FGQUADRATIC subtracts a quadratic amount from a group's membership as
%    a function of time that has passed since the group has been
%    last updated.  User-definitions include the time interval and the
%    amount of membership to subtract.  This function causes the forget
%    rate to be slow for short time differences and increase quickly as the
%    time difference increases.
%
%  Input
%
%    region - A list of groups for a given region in the memory
%    data structure, which is defined and documented in 'motionInterest.m'.
%
%    h - The index into the variable 'region' to get the exact group.
%
%    sTime - The current serial date and time of the video.  This variable
%    is compared against the date stored within the group.
%
%  Output
%
%    region - The same data structure given as input.  However, this
%    output may have one less groups due to forgetting.
%
%    del - Will be equal to zero if the group was not deleted.  Will be
%    equal to one if the group was deleted.
%
%  Examples
%
%    x = 3;
%    sTime = datenum('19-Feb-2008 00:00:10');
%    [data,del] = fgQuadratic( data, x, sTime );

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

%%%%%%%%%%%%%%%%%%%%%%%%%
% User-definitions
%%%%%%%%%%%%%%%%%%%%%%%%%

% Days to forget a standard membership unit.
days = 7;

% Amount of membership to forget, which is the standard membership unit.
fm = 0.05;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Perform the membership subtraction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Slope is defined by the membership over the days squared
slope = -fm / days^2;

% Quadratic forget funcion (y = ax^2 + b):
%   y -> new membership level
%   a -> quadratic slope/coefficient
%   x -> absolute delta time (sTime should be greater)
%   b -> old membership level
region.m(h) = (slope * abs(sTime - region.f(h))^2) + region.m(h);

% Assume we will not delete this group
del = 0;

% Remove group if no membership
if region.m(h) <= 0

  region.r(h,:) = [];
  region.g(h,:) = [];
  region.b(h,:) = [];
  region.h(h,:) = [];
  region.w(h,:) = [];
  region.vx(h,:) = [];
  region.vy(h,:) = [];
  region.t(h,:) = [];
  region.m(h) = [];
  region.d(h) = [];
  region.f(h) = [];
  region.i(h) = [];

  % We deleted
  del = 1;

else

  % Set the last update date/time flag
  region.f(h) = sTime;

end

function [data] = checkgroups(data,ind,ver)
% CHECKGROUPS Merge groups if discovered to be similar.
%
%  Syntax
%
%    data = checkgroups(data,ind,ver);
%
%  Description
%
%    CHECKGROUPS checks if the provided groups should remain separate or be
%    merged together because they are similar.  There are two ways to
%    determine if two groups are similar.  First, if the mean of one group
%    has higher membership in the other group, then the groups shall be
%    merged.  Second, if each group mean is within a user-defined
%    confidence range of the other group, then the groups shall be merged.
%    If groups are merged, the returned data structure will have fewer
%    groups.
%
```

```
%  Input
%
%    data - One region of the regional memory data structures, which
%    is defined and documented in 'motionInterest.m'.
%
%    ind - A one dimensional list of the indices of the groups in the
%    region to determine if they should be merged.
%
%    ver - Shall be equal to 'motion' if looking for similar motion events
%    and 'still' if looking for similar still events.
%
%  Output
%
%    data - The same region of the regional memory data structures.
%    However, this output may have fewer groups if any merging took place.
%
%  Examples
%
%    ind = [1 2 16 99];
%    data = checkgroups(data,ind,'motion');

% Jeremy Paskali : Ver 0.0 : 09-10-2006
% Jeremy Paskali : Ver 1.0 : 11-01-2006
% Jeremy Paskali : Ver 1.3 : 01-15-2007

% Set the user-defined confidence level of merging similar groups.
% This percentage is a floating point value from 0 to 1, inclusive.
confidence = 0.75;

i = 1;
while i <= numel(ind)
  j = i + 1;
  while j <= numel(ind)

    % Store indices
    g = ind(i);
    h = ind(j);

    check1 = 0;
    check2 = 0;

    % Groups should be merged if:
    % 1. Mean of one group has higher membership in the other group.
    % - OR -
    % 2. Group similaries pass the user-defined confidence level going in
    %    in both directions.

    % Calculate mean differences
    dr = abs( data.r(h,1) - data.r(g,1) );
    dg = abs( data.g(h,1) - data.g(g,1) );
    db = abs( data.b(h,1) - data.b(g,1) );
    dh = abs( data.h(h,1) - data.h(g,1) );
    dw = abs( data.w(h,1) - data.w(g,1) );

    if strcmp(ver,'motion')
      dvx = abs( data.vx(h,1) - data.vx(g,1) );
      dvy = abs( data.vy(h,1) - data.vy(g,1) );
```

```
end

dta = abs( data.t(h,1) - data.t(g,1) );

% Check for Sat to Sun rollover
% This is where h is on Sun and g is on Sat
% one week = 168 hours
dtb = data.t(h,1) - (data.t(g,1) - 168);

% This is where h is on Sat and g is on Sun
dtc = (data.t(g,1) + 168) - data.t(h,1);

% Get the minimum time as a check for both the Sat to Sun rollover
% and for normal time operations during the week
if dta <= dtb && dta < dtc

  % Normal time (No rollover)
  dt = dta;
  roll = 0;

elseif dtb <= dta && dtb <= dtc

  % h is on Sun and g is on Sat
  dt = dtb;
  roll = 1;

else

  % h is on Sat and g is on Sun
  dt = dtc;
  roll = 2;

end

% Calculate membership of g in h
mr = -(data.m(h)/data.r(h,2))*dr + data.m(h);
mg = -(data.m(h)/data.g(h,2))*dg + data.m(h);
mb = -(data.m(h)/data.b(h,2))*db + data.m(h);
mh = -(data.m(h)/data.h(h,2))*dh + data.m(h);
mw = -(data.m(h)/data.w(h,2))*dw + data.m(h);

if strcmp(ver,'motion')
  mvx = -(data.m(h)/data.vx(h,2))*dvx + data.m(h);
  mvy = -(data.m(h)/data.vy(h,2))*dvy + data.m(h);
end

mt = -(data.m(h)/data.t(h,2))*dt + data.m(h);

% Average membership of g in h
if strcmp(ver,'motion')
  member = mean( [mr mg mb mh mw mvx mvy mt], 2 );
else
  member = mean( [mr mg mb mh mw mt], 2 );
end

% If g has greater membership in h than itself
if member > data.m(g)
```

```
  % Switch roll because g is merging into h
  if roll == 1
    roll = 2;
  elseif roll == 2
    roll = 1;
  end

  % Merge g into h
  data = mergegroups( data, g, h, roll );

  % alter indices accordingly
  ind(i) = [];
  ind(i:end) = ind(i:end) - 1;

  continue;
end

% Test for similarity of g in h
test = member / data.m(h);

% g is at least this percentage similar to h
if test >= confidence
  check1 = 1;
end

% Calculate membership of h in g
mr = -(data.m(g)/data.r(g,2))*dr + data.m(g);
mg = -(data.m(g)/data.g(g,2))*dg + data.m(g);
mb = -(data.m(g)/data.b(g,2))*db + data.m(g);
mh = -(data.m(g)/data.h(g,2))*dh + data.m(g);
mw = -(data.m(g)/data.w(g,2))*dw + data.m(g);

if strcmp(ver,'motion')
  mvx = -(data.m(g)/data.vx(g,2))*dvx + data.m(g);
  mvy = -(data.m(g)/data.vy(g,2))*dvy + data.m(g);
end

mt = -(data.m(g)/data.t(g,2))*dt + data.m(g);

% Discover membership of h in g
if strcmp(ver,'motion')
  member = mean( [mr mg mb mh mw mvx mvy mt], 2 );
else
  member = mean( [mr mg mb mh mw mt], 2 );
end

% If h has greater membership in g than itself
if member > data.m(h)

  % Merge h into g
  data = mergegroups( data, h, g, roll );

  % alter indices accordingly
  ind(j) = [];
  ind(j:end) = ind(j:end) - 1;
```

```
      continue;
    end

    % Test for similarity of g in h
    test = member / data.m(g);

    % h is at least this percentage similar to g
    if test >= confidence
      check2 = 1;
    end

    % Check if both means are within variances
    if check1 == 1 && check2 == 1

      % Extremely similar groups.
      % Merge the lower membership into the larger.
      if data.m(g) > data.m(h)

        % Merge h into g
        data = mergegroups( data, h, g, roll );

        % alter indices accordingly
        ind(j) = [];
        ind(j:end) = ind(j:end) - 1;

      else

        % Merge g into h
        data = mergegroups( data, g, h, roll );

        % alter indices accordingly
        ind(i) = [];
        ind(i:end) = ind(i:end) - 1;

      end

    end

    j = j + 1;

  end

  i = i + 1;

end
```

Bibliography

[1] Matej Artac, Matjaz Jogan, and Ales Leonardis. Incremental pca for on-line visual learning and recognition. In *Proceedings of the 16th International Conference on Pattern Recognition*, 2002.

[2] Sergei Arthur, David; Vassilvitskii. On the worst case complexity of the k-means method. Technical report, Standford University, November 2005.

[3] Sang-Woo Ban and Minho Lee. Selective attention-based novelty scene detection in dynamic environments. *Neurocomputing*, 69:1723–1727, 2006.

[4] A. Billard. Machine learning techniques and applications, 2004. Ecole Polytechnique Federale De Lausanne.

[5] Paul S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. Technical report, Microsoft Research, One Microsoft Way Redmond, WA 98052, May 1998.

[6] David Burlone. A biologically plausible system for detecting saliency in video. Master's thesis, Rochester Institute of Technology, May 2006.

[7] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. Technical report, University of Karlsruhe, Germany, 2004.

[8] C. Clavel, T. Ehrette, and G. Richard. Events detection for an audio-based surveillance system. Technical report, Thales Research and Technology France, 2005.

[9] Mohamed Dahmane and Jean Meunier. Real-time video surveillance with self-organizing maps. In *Proceedings of the Second Canadian Conference on Computer and Robot Vision*, 2005.

[10] R. Gaborski, A. Vaingankar, V. Chaoji, and A. Teredesai. Venus: A system for novelty detection in video streams with learning. In *Proceedings of the 17th International FLAIRS Conference*, South Beach, FL, USA, May 2004.

[11] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Proceedings 17th NIPS*, 2003.

[12] Patricio T. Huerta, Linus D. Sun, Matthew A. Wilson, and Susumu Tonegawa. Formation of temporal memory requires nmda receptors within ca1 pyramidal neurons. *Neuron*, 25:473–480, 2000.

[13] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[14] Thomas W. James and Isabel Gauthier. Brain areas engaged during visual judgments by involuntary access to novel semantic information. *Vision Research*, 44:429–439, 2004.

[15] Kent A. Kiehl, Kristin R. Laurens, Timothy L. Duty, Bruce B. Forster, and Peter F. Liddle. An event-related fmri study of visual and auditory oddball tasks. *Journal of Psychophysiology*, 15:221–240, 2001.

[16] K. Krishna and M. Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 29(3):433–439, June 1999.

[17] Manfred Laucht, Katja Becker, and Martin H. Schmidt. Visual exploratory behaviour in infancy and novelty seeking in adolescence: two developmentaly specific phenotypes of drd4? *Journal of Child Psychology and Psychiatry*, 47:11:1143–1151, 2006.

[18] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, March 2003.

[19] Stephen Marsland. On-line novelty detection. University of Manchester.

[20] Mathworks. *Matlab Release 14 with Service Pack 3 Help*, August 2005.

[21] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.

[22] Hugo Vieira Neto and Ulrich Nehmzow. Automated exploration and inspection: Comparing two visual novelty detectors. In *Advanced Robotic Systems International*, 2005.

[23] Jeremy Paskali. Relating interest amongst observed events: An independent study. Rochester Institute of Technology, August 2006.

[24] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[25] Dragoljub Pokrajac, Vesna Zeljkovic, and Longin Jan Latecki. Noise-resilient detection of moving objects based on spatial-temporal blocks. Technical report, Delaware State University and Temple University, 2005.

[26] Jose C. Principe, Neil R. Euliano, and W. Curt Lefebvre. *Neural and Adaptive Systems*. John Wiley & Sons, 2000.

[27] David Stawski and Jeremy Paskali. Somja: Self-organizing map java application. Rochester Institute of Technology, February 2006.

[28] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[29] Bryan A. Strange, Andrew Duggins, William Penny, Raymond J. Dolan, and Karl J. Friston. Information theory, novelty and hippocampal responses: unpredicted or unpredictable? *Neural Networks*, 18:225–230, 2005.